Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus

Radu Curticapean* Dániel Marx†

October 14, 2015

Abstract

By now, we have a good understanding of how NP-hard problems become easier on graphs of bounded treewidth and bounded cliquewidth: for various problems, matching upper bounds and conditional lower bounds describe exactly how the running time has to depend on treewidth or cliquewidth. In particular, Fomin et al. (2009, 2010) have shown a significant difference between these two parameters: assuming the Exponential-Time Hypothesis (ETH), the optimal algorithms for problems such as MAX CUT and EDGE DOMINATING SET have running time $2^{O(t)}n^{O(1)}$ when parameterized by treewidth, but $n^{O(t)}$ when parameterized by cliquewidth.

In this paper, we show that a similar phenomenon occurs also for counting problems. Specifically, we prove that, assuming the counting version of the Strong Exponential-Time Hypothesis (#SETH), the problem of counting perfect matchings

- has no $(2-\varepsilon)^k n^{O(1)}$ time algorithm for any $\varepsilon > 0$ on graphs of treewidth k (but it is known to be solvable in time $2^k n^{O(1)}$ if a tree decomposition of width k is given), and
- has no $O(n^{(1-\varepsilon)k})$ time algorithm for any $\varepsilon > 0$ on graphs of cliquewidth k (but it can be solved in time $O(n^{k+1})$ if a k-expression is given).

A celebrated result of Fisher, Kasteleyn, and Temperley from the 1960s shows that counting perfect matchings in planar graphs is polynomial-time solvable. This was later extended by Gallucio and Loebl (1999), Tesler (2000) and Regge and Zechina (2000) who gave $4^k \cdot n^{O(1)}$ time algorithms for graphs of genus k. We show that the dependence on the genus k has to be exponential: assuming #ETH, the counting version of ETH, there is no $2^{o(k)} \cdot n^{O(1)}$ time algorithm for the problem on graphs of genus k.

*Simons Institute for the Theory of Computing, Berkeley, and Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary, curticapean@cs.uni-sb.de

†Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), dmarx@cs.bme.hu. Research supported by ERC Grant PARAMTIGHT (No. 280152) and OTKA grant NK105645.

1 Introduction

Many NP-hard optimization problems are solvable in polynomial time when restricted to graphs of bounded treewidth. This fact is exploited in a wide range of contexts, perhaps most notably, in the design of parameterized algorithms and approximation schemes. There has been a significant amount of research on developing and improving algorithms for bounded-treewidth graphs, as well as on trying to understand the limitations of treewidth-based algorithms. Courcelle's Theorem [25, 26] is a very general result showing that if a problem can be formulated in a logical language called MSO_2, then it can be solved in linear-time on graphs of bounded treewidth; however, problem-specific techniques usually give more efficient algorithms that have better dependence on treewidth.

Thanks to a series of recent results, by now we have a fairly good understanding on how the running time has to depend on the treewidth of the input graph. Concerning upper bounds, the development of new algorithmic techniques, such as fast subset convolution [6, 88], *Cut & Count* [36, 76], and rank-based dynamic programming [8, 34, 53], improved the running times for various problems, sometimes in unexpected ways. On the other hand, matching conditional lower bounds were obtained for many of these problems [9, 12, 35, 36, 71, 72]. For example, fast subset convolution can be used to solve DOMINATING SET in time $3^k n^{O(1)}$ if a tree decomposition of width k is given [88], but there is no such algorithm with running time $(3-\varepsilon)^k n^{O(1)}$ for any $\varepsilon > 0$, assuming the Strong Exponential-Time Hypothesis (SETH) [71]. This hypothesis, introduced by Impagliazzo, Paturi, and Zane [61, 62], can be informally stated as the assertion that n-variable m-clause CNF-SAT cannot be solved in time $(2-\varepsilon)^n m^{O(1)}$ for any $\varepsilon > 0$. While the validity of this hypothesis is not accepted as widely in the community as some other complexity conjectures, SETH seems to be a very fruitful working assumption that explains why certain well-known algorithms are best possible and cannot be improved further [1–4, 10, 11, 15, 33, 71, 75, 90].

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

Cliquewidth. The notion of cliquewidth was introduced by Courcelle and Olariu [29] and can be seen as a generalization of treewidth that keeps some of the favorable algorithmic properties of bounded-treewidth graphs. The main motivation for this width measure lies in the observation that highly homogeneous structures of large treewidth, such as cliques and complete bipartite graphs, do not pose great difficulties for, e.g., INDEPENDENT SET, DOMINATING SET, or VERTEX COLORING. This homogeneity is then captured by so-called k-expressions: For a graph G, this is a construction scheme that successively builds G from graphs whose vertices are labeled by $1,\dots,k$ such that vertices of the same label cannot be distinguished in later steps. A graph has cliquewidth at most k if it has a k-expression.

Courcelle, Makowsky, and Rotics [27] generalized, to some extent, Courcelle's Theorem to bounded-cliquewidth graphs, but this generalization comes at a price: it gives linear-time algorithms only for problems defined by MSO_1 formulas, which is a proper subset of MSO_2 that does not allow quantification over *edge* sets. Many problems, such as HAMILTONIAN CYCLE, are (provably) not definable in MSO_1 and there is a large literature on designing algorithms for problems on bounded-cliquewidth graphs with problem-specific approaches [48, 56–58, 67, 68, 73, 79, 81, 89]. Unlike for problems parameterized by treewidth, most of these problems are not known to be fixed-parameter tractable parameterized by cliquewidth, that is, they can be solved in time $n^{f(k)}$, but not in time $f(k)n^{O(1)}$. Fomin et al. [50, 51] showed conditional lower bounds suggesting that this is not a shortcoming of algorithm design, but an inevitable price one has to pay for the generalization to cliquewidth.

Counting perfect matchings. Some of the algorithmic results mentioned above can be generalized to the counting versions of the problems, where the task is to count the number of solutions. For example, Courcelle's Theorem and its variant for cliquewidth have counting analogs [28]. However, it is a well-known fact that a counting problem can be significantly harder than its decision version: finding a perfect matching is a classic polynomial-time solvable problem [47], but a seminal result of Valiant showed that counting the number of perfect matchings is #P-hard [85], even in bipartite graphs, where this problem is equivalent to evaluating the permanent of a matrix. In the present paper, we show that problems such as counting perfect matchings are also amenable to the study of quantitative lower bounds outlined in the previous paragraphs: We obtain tight upper and lower bounds on the running time needed to count perfect matchings when parameterized by treewidth, cliquewidth or genus.

Parameterizing by treewidth. The counting version of Courcelle's Theorem [28] immediately shows that the problem #PERFMATCH of counting perfect matchings is fixed-parameter tractable parameterized by treewidth. Furthermore, standard dynamic programming techniques directly give a $3^k n^{O(1)}$ time algorithm. The base of the exponential part was improved by van Rooij et al. [88] using the technique of fast subset convolution:

THEOREM 1.1. ([88]) *The problem* #PERFMATCH *can be solved in time $2^k n^{O(1)}$ on an n-vertex graph G if a tree decomposition of width k is given.*

Our first result gives a matching lower bound: assuming #SETH (the natural counting analog of SETH, which a priori is a weaker hypothesis than SETH), the base of the exponent cannot be improved any further.

THEOREM 1.2. *Assuming* #SETH, *there is no algorithm that, given an n-vertex graph G together with a tree decomposition of width k, solves* #PERFMATCH *in time $(2-\varepsilon)^k n^{O(1)}$ for some fixed $\varepsilon > 0$.*

Our proof in fact gives a lower bound for the problem parameterized by cutwidth, a parameter that is an upper bound on treewidth and also on the related notion of pathwidth. Therefore, our proof implies Theorem 1.2, even when treewidth is replaced by pathwidth.

Parameterizing by cliquewidth. Besides the general algorithmic result on counting problems definable in MSO_1, counting problems for graphs of bounded cliquewidth were investigated mostly in the context of computing graph polynomials [57, 58, 73]. Makowsky et al. [73] showed that, given a k-expression of an n-vertex graph G, the matching polynomial can be computed in time $O(n^{2k+1})$; in particular, this gives an $O(n^{2k+1})$ time algorithm for #PERFMATCH. In the full version of this paper, we improve this algorithm:

THEOREM 1.3. *The problem* #PERFMATCH *can be solved in time $O(n^{k+1})$ on an n-vertex graph G if a k-expression of G is given in the input.*

Makowsky et al. [73] asked whether #PERFMATCH (or computing the matching polynomial) is fixed-parameter tractable parameterized by cliquewidth. We show that, assuming #SETH, the algorithm of Theorem 1.3 is optimal, up to constant additive terms in the exponent.

THEOREM 1.4. *There exists a constant $c^* \in \mathbb{N}$ such that the following holds: Assuming* #SETH, *there is no integer $k \geq 1$ such that* #PERFMATCH *can be solved in time $O(n^{k-c^*})$ on n-vertex graphs G that are given together with a k-expression.*

Thus in particular #PERFMATCH is not fixed-parameter tractable (assuming #SETH), and this holds also for the more general problem of computing the matching polynomial, answering the question of Makowsky et al. [73]. Our proof also shows that the problem is W[2]-hard.

**Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.**

Parameterizing by crossing number or genus. Finally, we turn our attention to graphs that are almost planar. A celebrated result of Fisher, Kasteleyn, and Temperley [63, 64, 82] showed that #PERFMATCH is polynomial-time solvable on planar graphs. If a graph is not planar, but a drawing with k crossings is given, then a simple branching algorithm can reduce the problem to 2^k planar instances of #PERFMATCH.[1]

THEOREM 1.5. *Given an n-vertex planar graph G with a drawing in the plane with k crossings,* #PERFMATCH *can be solved in time* $2^k n^{O(1)}$.

Thus the problem is polynomial-time solvable (and actually fixed-parameter tractable) for graphs of bounded crossing number; we avoid the discussion of how to find the drawing. More generally, Gallucio and Loebl [55], Tesler [83], and Regge and Zecchina [80] showed that the problem is fixed-parameter tractable even for graphs of bounded genus.

THEOREM 1.6. ([55, 80, 83]) *Given an n-vertex graph G embedded on a surface of genus g, the problem* #PERFMATCH *can be solved in time* $4^g n^{O(1)}$.

The base of the exponential is worse in Theorem 1.6 than in Theorem 1.5, which raises the obvious question of what the best possible base could be. While we cannot answer this question as tightly as in Theorems 1.2 and 1.4, we can at least show that the dependence on the parameter has to be exponential.

THEOREM 1.7. *Assuming* #ETH*, there is no* $2^{o(k)} n^{O(1)}$ *time algorithm for* #PERFMATCH *when given as input an n-vertex graph G and a drawing of G in the plane with k crossings.*

As the crossing number of a graph is always an upper bound on its genus, this theorem also gives a lower bound for the problem parameterized by genus.

Somewhat interestingly, this opposes a certain "square root phenomenon" that has been observed in the context of parameterized algorithms on planar graphs: for many decision problems, the best possible running times (assuming ETH) are often of the form $2^{O(\sqrt{k})} n^{O(1)}$ or $n^{O(\sqrt{k})}$ [24, 38–46, 52, 54, 65, 66, 77, 78, 84]. Thus one could have expected that the running time has a dependence of the form $2^{O(\sqrt{k})}$ on the number k of crossings introduced into the planar drawing. However, Theorem 1.7 shows that this is not the case: the dependence has to be single-exponential. Similar violations of the square root phenomenon for counting problems have also been observed for #PERFMATCH on k-apex graphs, where a $n^{\Omega(k/\log k)}$ lower bound is known [32].

Reductions among counting problems. If the decision version of a problem is polynomial-time solvable, then hardness of the counting version *cannot* be proved by a parsimonious reduction from an NP-hard problem, as this would imply that the decision version of the NP-hard problem is polynomial-time solvable. Therefore, such hardness results necessarily involve a "mysterious" step that is highly non-parsimonious and usually very specific to the particular reduction source and target.

A standard way of doing this for #PERFMATCH introduces a weighted version of the problem: each edge is equipped with a weight, the weight of a perfect matching is defined as the *product* of the weights of its edges, and the task is to compute the sum of the weights of all the perfect matchings. Crucially, weights can be negative, thus allowing different perfect matchings to cancel out each other. After proving hardness for the weighted version, the negative weights can be eliminated by modular arithmetic [85] or polynomial interpolation [7, 30, 37], yielding hardness of the unweighted problem under Turing reductions. It is precisely at this step that the reduction becomes non-parsimonious: the existence of a perfect matching in one of the constructed instances does not tell directly whether the source instance has a solution or not.

While this technique is very powerful, we do not want to repeat it all over again every time a lower bound is proved for some counting problem. Therefore, we hide these steps behind a layer of abstraction, the so-called Holant framework: We first prove hardness for a certain type of Holant problem and then reduce this to #PERFMATCH in a second step. This last step encapsulates the arguments involving negative weights and polynomial interpolation, which allows us focus on our main task: constructing instances with bounded treewidth or cliquewidth. While the Holant framework introduces some notational overhead, the proofs become relatively transparent and comparable to similar lower bounds for decision problems.

The Holant framework. Let us briefly present the Holant framework (see Section 3 for more details), which was introduced by Valiant in the context of holographic algorithms [86], and which was extended into a more general framework since then [13, 15–20, 59, 60, 69].

A *signature graph* is a graph Ω with a weight $w(e)$ on each edge and a function $f_v : \{0,1\}^{I(v)} \to \mathbb{Q}$ associated with each vertex v, where $I(v)$ is the set of edges incident to v. That is, f_v assigns a rational value to every subset of edges of $I(v)$; we call f_v the *signature* of v. Then for every subset $x \in \{0,1\}^{E(\Omega)}$ of edges, the functions f_v determine a rational value at each vertex v in a natural way. We define the *value* of x to be the product of these values for

[1]The basic idea is that we select one edge from each of the k crossings and branch on which of the 2^k possible subsets of these k edges appear in the perfect matching. If an edge does not appear in the matching, then we remove it; otherwise, we remove its endpoints. In all cases, we get a planar graph.

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

all $v \in V(\Omega)$ and we define the *weight* of a subset of edges to be the product of the weights of the edges. The Holant of Ω sums up, for every subset x of edges, the product of the weight of x and the value of x:

$$\mathrm{Holant}(\Omega) = \sum_{x \in \{0,1\}^{E(\Omega)}} \prod_{e \in x} w(e) \prod_{v \in V(\Omega)} f_v(x|_{I(v)}).$$

To approach this somewhat abstract problem, it may help to observe that it contains counting perfect matchings as a special case. Let G be a graph, define $\Omega := G$, and assign $w(e) = 1$ to every $e \in E(\Omega)$. For every $v \in V(\Omega)$, define f_v to be 1 if exactly one edge of $I(v)$ is selected and 0 otherwise. In this case, we also say that f_v has the signature $\mathtt{HW}_{=1}$, which is short for "Hamming weight = 1". Then $\mathrm{Holant}(\Omega)$ counts exactly those edge subsets $x \subseteq E(G)$ where $f_v(x) \neq 0$ at every vertex v, that is, where every vertex has degree exactly one in x. Therefore, one can think of Holant problems as a certain generalization of counting weighted perfect matchings and other degree-bounded subgraph counting problems.

In our proofs of lower bounds based on the Holant framework, we first reduce the problem of counting the satisfying assignments of a CNF-SAT formula to that of computing the Holant of a certain signature graph Ω. Then we transform Ω to a signature graph Ω' in which every vertex has signature $\mathtt{HW}_{=1}$. This can equivalently be viewed as a weighted version of #PERFMATCH. Finally, a self-contained weight removal step shows that $\mathrm{Holant}(\Omega')$ can be reduced to unweighted #PERFMATCH. The transformation from Ω to Ω' is performed using certain gadgets: if a vertex v of degree d features some signature f_v, then we replace it with a signature graph that has d external edges and (in a well-defined sense) computes exactly the function f_v. The Holant framework provides a natural language for a rigorous treatment of this operation.

For our purposes, gadgets using only the signature $\mathtt{HW}_{=1}$ are of particular interest; such gadgets are known as *matchgates* in the literature [14, 87]. It should be noted here that the established meaning of the term "matchgate" refers to planar gadgets, whereas in this paper, we deviate from this convention by explicitly allowing non-planar matchgates as well. This allows us to make the crucial observation that *every* signature of constant size (satisfying a trivially necessary parity condition) can be realized by a matchgate of constant size. Therefore, for our purposes, it is sufficient to construct a signature graph Ω where every vertex has the signature $\mathtt{HW}_{=1}$ *or* has bounded degree. Then the rest of the reduction from $\mathrm{Holant}(\Omega)$ to #PERFMATCH is completely automatic, we only need to verify that the these steps change treewidth or cliquewidth of the graph in a controlled way (but usually this is easy). We encapsulate this reduction from Holant problems to #PERFMATCH in Theorem 4.1. The hardness proofs in Theorems 1.2, 1.4, and 1.7 all construct Holant instances and then invoke Theorem 4.1 to complete the reduction to #PERFMATCH. We believe that this way of approaching hardness results for counting problems would also be fruitful for other problems and parameters.

2 Preliminaries

2.1 Complexity assumptions. The *Exponential Time Hypothesis* (ETH) conjectured by Impagliazzo, Paturi and Zane [62] and its strong variant (SETH) are conjectures about the exponential time complexity of k-SAT. Let s_k be the infimum over all δ such that n-variable k-SAT can be solved in $2^{\delta n}$ time. Then ETH states that $s_3 > 0$. A simple and perhaps more intuitive consequence of ETH is that there is no $2^{o(n)}$ time algorithm for n-variable 3SAT, that is, no algorithm for 3SAT is subexponential in the number of variables. On its own, this may not rule out algorithms that are subexponential in the input size: the number of clauses can be superlinear in the number of variables. However, the Sparsification Lemma shows that ETH in fact rules out such algorithms as well. One way to formulate this result is the following:

LEMMA 2.1. ([62]) *Assuming* ETH*, n-variable m-clause* 3SAT *has no $2^{o(n+m)}$ time algorithm.*

Since CNF-SAT with n variables and m clauses has a $2^n \cdot m^{O(1)}$ time algorithm, the sequence $\{s_k\}_{k \in \mathbb{N}}$ is bounded from above by 1. Impagliazzo and Paturi [61] conjecture that 1 is indeed the limit of this sequence, a statement that became later known as SETH, the Strong Exponential Time Hypothesis [23]. We can formulate a convenient consequence of SETH by saying that n-variable m-clause CNF-SAT has no $(2-\varepsilon)^n m^{O(1)}$ time algorithm for any $\varepsilon > 0$.

In the context of counting problems, it is natural to use the counting versions of ETH and SETH, which are defined analogously in terms of the counting problems #3SAT and #CNF-SAT (see [37]). Clearly, #ETH and #SETH are weaker assumptions than ETH and SETH, respectively, as they only rule out the existence of improved counting algorithms. Thus stating negative results assuming #ETH instead of ETH makes the result a priori stronger, but perhaps more importantly, it seems very natural and closer to the spirit of the problem to start our reductions from genuine counting problems.

Dell et. al [37] showed that the Sparsification Lemma can be made to work also for the counting version. Thus we have the following counting analog of Lemma 2.1.

LEMMA 2.2. ([62]) *Assuming* #ETH*, the problem* #3SAT *on formulas with n variables and m clauses cannot be solved in time $2^{o(n+m)}$.*

Analogously to SETH, assuming #SETH has the consequence that n-variable m-clause #CNF-SAT has no $(2-\varepsilon)^n \cdot m^{O(1)}$ time algorithm for any $\varepsilon > 0$.

**Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.**

2.2 Inserting gadgets into graphs. Our graph notation is standard, but we need to introduce formal definitions for replacing vertices with gadgets in graphs. For this purpose, we find it convenient to use the notion of *dangling edges,* which are edges having only one endpoint [22].

Let H_v be a graph with d distinguished dangling edges $e_1,\dots,e_d$ incident to distinct *portal vertices* $v_1,\dots,v_d$, respectively. Let G be a graph, let $v \in V(G)$ be of degree d, and assume that an ordering f_1, …, f_d of the d edges incident to v is given. Then the operation of *inserting H_v at v* is defined as follows: take the disjoint union of G and H_v, remove v, and for every $1 \le i \le d$, remove e_i and replace it with an edge connecting v_i with the other endpoint of f_i.

We can extend this operation to inserting more than one graph in parallel. That is, assume we are given vertices $v^1,\dots,v^t$ in G (each vertex having a fixed ordering of the edges incident to it) and graphs $H_{v^1},\dots,H_{v^t}$ such that the number of dangling edges of H_{v^i} and the degree of v^i agree, we can define the insertion of H_{v^i} at v^i in parallel for $1 \le i \le t$ with the obvious meaning depicted in Figure 1.

2.3 Treewidth, pathwidth and cutwidth. We recall the most important notions related to treewidth in this section. A *tree decomposition* of a graph G is a pair $(T,\mathcal{B})$ in which T is a tree and $\mathcal{B} = \{B_t \mid t \in V(T)\}$ is a family of subsets of $V(G)$ such that

1. $\bigcup_{t\in V(T)} B_t = V$;
2. for each edge $e = uv \in E(G)$, there exists a node $t \in V(T)$ such that $u,v \in B_t$, and
3. the set of nodes $\{t \in V(T) \mid v \in B_t\}$ forms a connected subtree of T for every $v \in V(G)$.

To distinguish between vertices of the original graph G and vertices of T in the tree decomposition, we call vertices i of T *nodes* and their corresponding B_i's *bags*. The *width* of the tree decomposition is the maximum size of a bag in $\mathcal{B}$ minus 1. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum width over all possible tree decompositions of G.

A *path decomposition* is a tree decomposition where T is a path. The *pathwidth* $\mathrm{pw}(G)$ of a graph G is the minimum width over all possible path decompositions of G. By definition, we have $\mathrm{tw}(G) \le \mathrm{pw}(G)$.

A *linear layout* of an n-vertex graph G is an ordering v_1, …, v_n of $V(G)$. For $1 \le i \le n$, the *cut after v_i* is the set of edges between $\{v_1,\dots,v_i\}$ and $\{v_{i+1},\dots,v_n\}$. The *cutwidth* of the linear layout is the maximum size of the cut after v_i for $1 \le i < n$. The *cutwidth* of G, denoted by $\mathrm{cutw}(G)$, is the minimum cutwidth over all possible linear layouts of G. A linear layout of cutwidth $\mathrm{cutw}(G)$ is called *optimal.* It is easy to see that $\mathrm{pw}(G) \le \mathrm{cutw}(G)$.

The following lemma shows that inserting gadgets of bounded cutwidth increases cutwidth only by a constant (this property of cutwidth is the main reason we are using it in our proofs).

LEMMA 2.3. *Let G be a graph and let $X \subseteq V(G)$ be a subset of vertices, each of degree at most d. For every $v \in X$, let us replace v by inserting a graph H_v of cutwidth at most c; let G' be the resulting graph. Then* $\mathrm{cutw}(G') = \mathrm{cutw}(G) + d + c$.

2.4 Cliquewidth. We follow Fomin et al. [51] for the basic definitions of cliquewidth. Let G be a graph and t be a positive integer. A *t-graph* is a graph with vertices labeled by integers from $\{1,2,\dots,t\}$. We refer to a t-graph consisting of exactly one vertex labeled by some $i \in \{1,2,\dots,t\}$ as an *initial t-graph.* The *cliquewidth* $\mathrm{cw}(G)$ is the smallest integer t such that G can be constructed by repeated applications of the following four operations:

$i(v)$: *Introduce* operation constructing an initial t-graph with vertex v labeled by i,

$\oplus$: *Disjoint union* of two t-graphs,

$\rho_{i\to j}$: *Relabel* operation changing all labels i to j, and

$\eta_{i,j}$: *Join* operation making all vertices labeled by i adjacent to all vertices labeled by j.

An *expression tree* of a graph G is a rooted tree T with nodes of four types i, $\oplus$, ρ, and η, corresponding to the operations described above. To each node, a t-graph is associated such that G is isomorphic to the graph corresponding to the root of T after removal of all labels.

Introduce nodes $i(v)$ are precisely the leaves of T, and each such node corresponds to an initial t-graph consisting of the i-labeled vertex v.

Union node $\oplus$ stands for a disjoint union of graphs associated with its children.

Relabel node $\rho_{i\to j}$ has one child and is associated with the t-graph obtained by the application of the relabeling operation to the graph corresponding to its child.

Join node $\eta_{i,j}$ has one child and is associated with the t-graph resulting from the application of the join operation to the graph corresponding to its child.

The *width* of the expression tree T is the number of different labels appearing in T. If G is of cliquewidth t, then there is a rooted expression tree T of width t for G. An expression tree T is *irredundant* if for any join node $\eta_{i,j}$, the vertices labeled by i and j are not adjacent in the graph associated with its child. It was shown by Courcelle and Olariu [29] that every expression tree T of G can be transformed into an irredundant expression tree T' of the same width in time linear in the size of T. We will use this for algorithmic purposes.

The following definitions related to clique expressions are nonstandard, but we find them very useful for

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

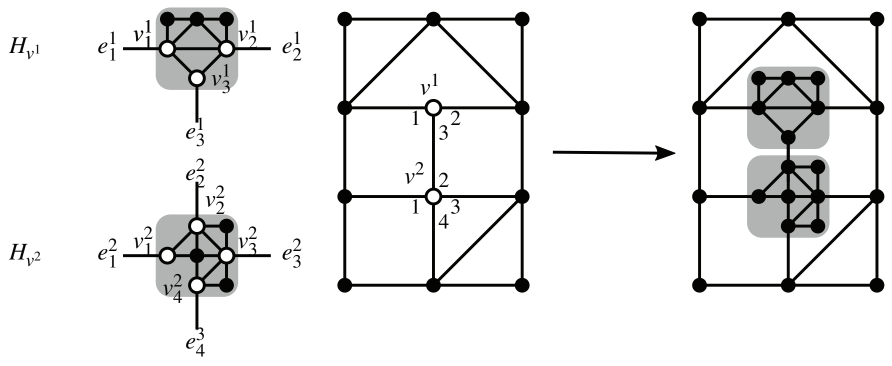


Figure 1: Inserting gadgets H_{v_1} and H_{v_2} at v_1 and v_2, respectively. The numbers around v_1 and v_2 show the ordering of the edges incident to them.

our purposes. We say that a label i is *singleton* if for every subexpression, the labeled graph associated to it contains at most one vertex with label i and there is no operation $\rho_{j\to i}$ in the expression (i.e., the only way a vertex can get label i is by an $i(v)$ operation). Sometimes we say that a label is *large* to emphasize that it is not required to be singleton. A (k,s)-*expression* is a $(k+s)$-expression with at least s singleton labels.

We say that an edge uv is *singular* in an irredundant clique expression if the unique operation $\eta_{i,j}$ creating uv created exactly one edge (i.e., there was exactly one vertex with label i and exactly one vertex with label j at this point). We say that label i is a *forget* label if there is no operation $i(v)$, $\rho_{i\to j}$, or $\eta_{i,j}$ in the k-expression for any j (thus label i may appear only in an operation of the form $\rho_{j\to i}$). We say that a vertex v is *singular* if every edge incident to v is singular and whenever the graph associated to a subexpression contains v, then v has either a singleton label or a forget label.

It is known and easy to see that $\text{cw}(G) \le \text{pw}(G) + 2$ [49] and we can verify that the proof provides a $(1, \text{pw}(G)+1)$-expression featuring only singular vertices. We prove that inserting bounded-pathwidth gadgets at singular vertices increases the cliquewidth only moderately: only the number of singleton labels is increased.

LEMMA 2.4. *Let G be a graph and let $X \subseteq V(G)$ be a subset of vertices, each of degree at most d. Suppose that G has a (k,s)-expression T where every $v \in X$ is singular. For every $v \in X$, replace v by inserting a graph G_v of pathwidth at most p; let G' be the resulting graph. Then* $\text{cw}(G') \le k + d(s+1) + p + 1$.

3 Holants and Matchgates

We give an introduction to what we call the *Holant framework*, a toolbox based on [21, 22, 87]. A more detailed exposition can be found in Chapter 2 of [31].

3.1 Holants.

Given a graph G and $v \in V(G)$, we denote the edges incident with v by $I(v)$.

DEFINITION 1. *A* signature graph *is an edge-weighted graph Ω, which may feature parallel edges, and which has a* signature $f_v : \{0,1\}^{I(v)} \to \mathbb{Q}$ *associated with each vertex $v \in V(\Omega)$.*

The *Holant* of Ω is a particular sum over the edge assignments $x \in \{0,1\}^{E(\Omega)}$. Given an assignment $x \in \{0,1\}^{E(\Omega)}$, we say that an edge $e \in E(\Omega)$ is *active in x* if $x(e) = 1$ holds, otherwise e is *inactive in x*. We tacitly identify x with the set of active edges in x. Given a subset $S \subseteq E(\Omega)$, we write $x|_S$ for the restriction of x to S, which is the unique assignment in $\{0,1\}^S$ that agrees with x on S.

DEFINITION 2. ([87]) *Let Ω be a signature graph with edge weights $w : E(\Omega) \to \mathbb{Q}$ and a function $f_v : \{0,1\}^{I(v)} \to \mathbb{Q}$ for each $v \in V(\Omega)$. Furthermore, let $x \in \{0,1\}^{E(\Omega)}$ be an assignment to the edges of Ω. Then we define*

$$\text{val}_\Omega(x) := \prod_{v\in V(\Omega)} f_v(x|_{I(v)}), \tag{3.1}$$

$$w_\Omega(x) := \prod_{e\in x} w(e), \tag{3.2}$$

and we say that x satisfies *Ω if $\text{val}_\Omega(x) \neq 0$ holds. Furthermore, we define*

$$\text{Holant}(\Omega) := \sum_{x\in\{0,1\}^{E(\Omega)}} w_\Omega(x) \cdot \text{val}_\Omega(x). \tag{3.3}$$

A particularly useful type of signatures is that of *Boolean functions*, whose ranges are restricted to $\{0,1\}$ rather than $\mathbb{Q}$. If all signatures appearing in Ω are Boolean, then $\text{Holant}(\Omega)$ simply sums over those assignments $x \in \{0,1\}^{E(\Omega)}$ that pass all constraints imposed by the vertex functions, and each x is weighted by $w_\Omega(x)$. We use the following Boolean signatures:

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

DEFINITION 3. *For $x \in \{0,1\}^*$, let $\text{hw}(x)$ be the Hamming weight of x, that is, the number of ones in x. For statements φ, we define $[\varphi] = 1$ if φ is true, and $[\varphi] = 0$ otherwise. For any arity $k \in \mathbb{N}$ and $x \in \{0,1\}^k$, we then define signatures*

$$\begin{aligned} \texttt{HW}_{=\texttt{d}}(x) &= [\text{hw}(x) = d], \qquad \textit{for } d \in \mathbb{N}, \\ \texttt{EVEN}(x) &= [\text{hw}(x) \textit{ even}], \\ \texttt{ODD}(x) &= [\text{hw}(x) \textit{ odd}]. \end{aligned}$$

3.2 Gates and matchgates. Given a signature graph Ω, we can sometimes simulate signatures by gadgets or *gates*, which are signature graphs with *dangling edges*. A dangling edge is an "edge" with only one endpoint. These notions are borrowed from the $\mathcal{F}$-gates in [22].

DEFINITION 4. *For disjoint sets A and B, and for assignments $x \in \{0,1\}^A$ and $y \in \{0,1\}^B$, we write $xy \in \{0,1\}^{A\cup B}$ for the assignment that agrees with x on A, and with y on B. We also say that the assignment xy* extends *x.*

A gate *is a signature graph Γ, possibly containing a set $D \subseteq E(\Gamma)$ of dangling edges, all of which have edge weight 1. A gate Γ is a* matchgate *if it features only the signature $\texttt{HW}_{=1}$. The* signature realized by Γ *is the function* $\text{Sig}(\Gamma) : \{0,1\}^D \to \mathbb{Q}$ *that maps x to*

$$\text{Sig}(\Gamma, x) = \sum_{\substack{xy\in\{0,1\}^{E(\Gamma)} \\ \text{extends}\, x}} w_\Gamma(xy) \cdot \text{val}_\Gamma(xy). \tag{3.4}$$

We consider the dangling edges D of gates Γ to be labeled as $1, \dots, |D|$. This way, we can consider signatures $\text{Sig}(\Gamma)$ *as functions of type $\{0,1\}^{|D|} \to \mathbb{Q}$ instead of $\{0,1\}^D \to \mathbb{Q}$.*

REMARK 1. *Note that we allow matchgates to be non-planar. This deviates from the notion of matchgates established in the literature [14, 87], which are required to be planar. More precisely, a matchgate on dangling edges $1, \dots, d$ is planar if it can be drawn in the plane without crossings such that its dangling edges appear in the order $1, \dots, d$ on its outer face.*

The usefulness of gates for our arguments lies in their ability of simulating complex signatures by simpler signatures. For instance, for any even arity $k \in \mathbb{N}$, we can realize the k-ary signature

$$\texttt{EQ}_k(x_1, \dots, x_k) := [x_1 = \dots = x_k]$$

by a gate whose vertices feature only the equality signature $\texttt{EQ}_4$ of arity 4.

EXAMPLE 1. *For all even $k \geq 4$, there exists a gate $\Gamma_{\texttt{EQ}}$ with $\text{Sig}(\Gamma_{\texttt{EQ}}) = \texttt{EQ}_k$. This gate consists of vertices $v_1, \dots, v_{k/2-1}$, each equipped with $\texttt{EQ}_4$, internal edges $e_1, \dots, e_{k/2-2}$ with $e_i = v_i v_{i+1}$ for all i, and k additional dangling edges, as shown in Figure 2.*

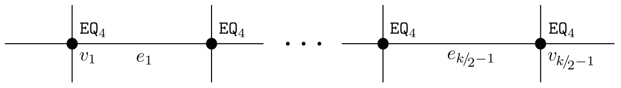


Figure 2: The gate $\Gamma_{\texttt{EQ}}$ with $\text{Sig}(\Gamma_{\texttt{EQ}}) = \texttt{EQ}_k$ in Example 1.

Its correctness can be verified from the definition: an assignment to the dangling edges of $\Gamma_{\texttt{EQ}}$ can be extended to a satisfying assignment for $\Gamma_{\texttt{EQ}}$ only if all values agree, in which case there is exactly one assignment.

In the following, we formalize the operation of *inserting* a gate Γ into a signature graph so as to simulate a vertex of a specific signature.

DEFINITION 5. *Let Ω be a signature graph, let $v \in V(\Omega)$ with $D = I(v)$ and let Γ be a gate with dangling edges D.*

Then we can insert Γ at v *by (i) deleting v and keeping D as dangling edges, and then (ii) placing a copy of Γ into Ω and identifying each dangling edge $e \in D$ across Γ and Ω. That is, if e has an endpoint u in Ω, and an endpoint v in Γ, then we consider e as an edge uv in the resulting graph.*

A simple calculation shows that insertions of suitable matchgates preserve Holants.

LEMMA 3.1. *Let Ω be a signature graph and let $v \in V(\Omega)$. Let Ω' be derived from Ω by inserting a gate Γ with* $\text{Sig}(\Gamma) = f_v$ *at v. Then* $\text{Holant}(\Omega) = \text{Holant}(\Omega')$.

We focus on *matchgates*, as defined above, since these allow us to reduce the problem of computing Holants to #PERFMATCH. Note that only graphs with an even number of vertices admit perfect matchings. This implies the following parity condition on signatures of matchgates.

FACT 3.1. (PARITY CONDITION) *If a signature $f : \{0,1\}^d \to \mathbb{Q}$ can be realized by a matchgate, then at least one of the following holds:*

- *For all $x \in \{0,1\}^d$ with odd* $\text{hw}(x)$, *we have $f(x) = 0$. Then we call f even.*
- *For all $x \in \{0,1\}^d$ with even* $\text{hw}(x)$, *we have $f(x) = 0$. Then we call f odd.*

We say that f is parity-consistent *if it is even or odd. We also say that a signature graph is parity-consistent if every vertex has a parity-consistent signature.*

Furthermore, we can summarize the use of matchgates in the following fact. To this end, given a graph G with edge weights $w : E(G) \to \mathbb{Q}$, let us define

$$\text{PerfMatch}(G) = \sum_M \prod_{e\in M} w(e),$$

where M ranges over the perfect matchings $M \subseteq E(G)$ of G. This was also defined in [87].

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

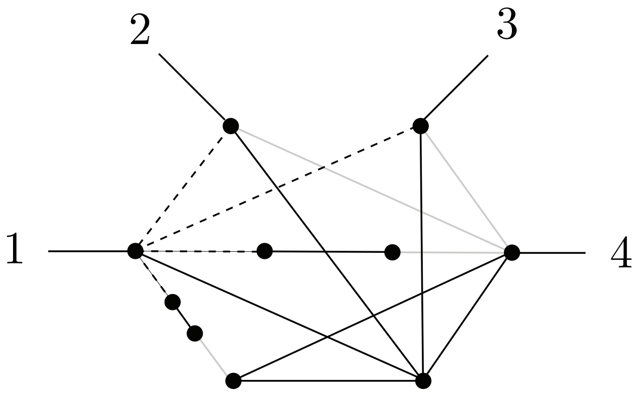


Figure 3: The equality matchgate $\Gamma_=$ that realizes $\texttt{EQ}_4$. Edges of weight -1, $1/2$ and 1 are shown gray, dashed and black, respectively. A comparable matchgate is shown in [7], but it realizes only a weighted version of the signature $\texttt{EQ}_4$.

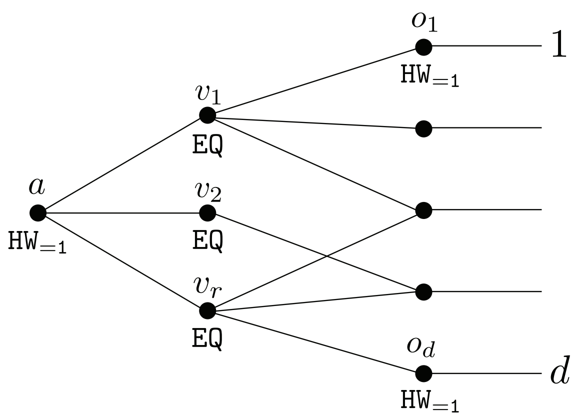


Figure 4: The gate Γ' constructed in the proof of Lemma 3.3. In this example, Γ' realizes the signature f that maps $\{00011, 11101, 11000\}$ to 1 and all other inputs to 0.

FACT 3.2. *Let Ω be a signature graph. If there is a matchgate Γ_v with $\mathrm{Sig}(\Gamma_v) = f_v$ for every vertex $v \in V(\Omega)$, then we can insert Γ_v as a gate at v, as specified in Definition 5. This yields a signature graph that uses only edge weights and the signature $\texttt{HW}_{=1}$. In other words, we obtain a graph $G = G(\Omega)$ on $\sum_v |V(\Gamma_v)|$ vertices and $\sum_v |E(\Gamma_v)|$ edges with*

$$\mathrm{Holant}(\Omega) = \mathrm{PerfMatch}(G(\Omega)).$$

3.3 Matchgates for parity-consistent signatures. As seen in Example 1, we can use matchgates for simple signatures to construct matchgates for more complex signatures. In the following, we use this to realize the signature $\texttt{EQ}_k$ for *even* $k \in \mathbb{N}$ from the matchgate $\Gamma_=$ for $\texttt{EQ}_4$ shown in Figure 3. We discovered $\Gamma_=$ using a computer algebra system, in a process that is detailed in Appendix C of [31]. Note that $\texttt{EQ}_k$ for odd $k \in \mathbb{N}$ does not satisfy the parity condition and hence cannot be realized by matchgates.

LEMMA 3.2. *For all even $k \in \mathbb{N}$, there is a matchgate realizing $\texttt{EQ}_k$. This matchgate features $O(k)$ vertices and edges and only $-1, \frac{1}{2}, 1$ as edge weights.*

Proof. For $k = 2$, we use a path on 4 vertices with dangling edges at its endpoints. For $k = 4$, we can verify that the matchgate $\Gamma_=$ in Figure 3 realizes $\texttt{EQ}_4$. For $k > 4$, we use the gate from Example 1 and realize each occurrence of $\texttt{EQ}_4$ by $\Gamma_=$.

The parity condition in Fact 3.1 tells us that every signature of a matchgate is even or odd. In the following, we show that the converse holds as well:

LEMMA 3.3. *Let $f : \{0,1\}^{[d]} \to I$ be a signature of arity $d \in \mathbb{N}$ that is parity-consistent. Then there is a matchgate Γ that realizes f. Furthermore, if $\mathrm{supp}(f)$ denotes the support of f, then*

- *Γ has $O(|\mathrm{supp}(f)| \cdot d)$ vertices and edges,*
- *Γ has maximum degree at most $|\mathrm{supp}(f)| + O(1)$,*
- *the edge weights of Γ are contained in the set $I \cup \{-1, 1/2, 1\}$,*
- *given as input $\{(x, f(x)) \mid x \in \mathrm{supp}(f)\}$, we can construct Γ in time $O(|\mathrm{supp}(f)| \cdot d)$.*

The construction of Γ resembles the construction of a formula in DNF from a given Boolean function. For each element $x \in \mathrm{supp}(f)$, we create an assignment gate L_x that tests whether the dangling edges of Γ are assigned x and this way ensures that $\mathrm{Sig}(\Gamma, x) = f(x)$. Furthermore, we ensure that $\mathrm{Sig}(\Gamma, x) \neq 0$ holds only if exactly one of these tests succeeds.

Proof. of Lemma 3.3. For this proof, we assume that $d - \mathrm{hw}(x)$ is odd for every $x \in \mathrm{supp}(f)$, which implies that exactly one of d and f is even, while the other one is odd. If $d - \mathrm{hw}(x)$ is even, the proof proceeds similarly.

We first define the following gate Γ' on dangling edges $[d]$, shown exemplarily in Figure 4. The matchgate Γ is then obtained by realizing all signatures appearing in Γ' by matchgates.

1. Create vertices $O = \{o_1, \ldots, o_d\}$ with signature $\texttt{HW}_{=1}$, and for $i \in [d]$, add the dangling edge i and make it incident to o_i.
2. Create a vertex a with signature $\texttt{HW}_{=1}$.
3. Let $\mathrm{supp}(f) = \{x_1, \ldots, x_r\}$ for some $r \in \mathbb{N}$. For each $\kappa \in [r]$, let $S_\kappa = O \setminus \{o_i \mid i \in x_\kappa\}$. Note that $|S_\kappa|$ is odd by assumption. We perform the following steps.
   (a) Create a vertex v_κ with signature $\texttt{EQ}_{|S_\kappa|+1}$ and make it adjacent to all vertices in S_κ. Note that

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

$|S_\kappa|+1$ is even, so $\mathtt{EQ}_{|S_\kappa|+1}$ can be realized by a matchgate by Lemma 3.3.

(b) Draw an edge of weight $f(x_\kappa)$ from v_κ to a.

We prove that Γ' realizes f. Let $y \in \{0,1\}^{E(\Gamma')}$ be a satisfying assignment. By $\mathtt{HW}_{=1}$ at the vertex a and $\mathtt{EQ}$ at v_κ for $\kappa \in [r]$, there is exactly one $\kappa \in [r]$ such that all edges of $I(v_\kappa)$ are active under y, while all edges in $I(v_{\kappa'})$ for $\kappa' \neq \kappa$ are inactive. In particular, we then have

$$\mathrm{val}_{\Gamma'}(y) \cdot w_{\Gamma'}(y) = f(x_\kappa). \tag{3.5}$$

Let $x = y|_{[d]}$ be the restriction of y to the dangling edges of Γ'. We observe that, if the edges in $I(v_\kappa)$ are active under y, for $\kappa \in [r]$, then $x = x_\kappa$: Since y is satisfying, by $\mathtt{HW}_{=1}$ at O, every $o_i \in O$ for $i \in [d]$ is incident with exactly one active edge, and this edge must be dangling if $x(i) = 1$, or contained in $I(v_k)$ if $x(i) = 0$. Hence, for every $x \in \{0,1\}^{[d]}$, there is a satisfying assignment y of Γ' that extends x if and only if $x = x_\kappa$ for some $\kappa \in [s]$. Furthermore, in this case, y is unique and satisfies (3.5). We conclude that $\mathrm{Sig}(\Gamma', x) = f(x)$.

4 From Holants to perfect matchings

In this section, we present our main technical tool that encapsulates the reduction from Holant problems to unweighted #PERFMATCH. From the previous section, we know that bounded-arity signatures can be replaced with matchgates that may feature edge weights (which are possibly even negative). In the following, we show how to simulate these weights in a careful way to ensure an overall bound on the cutwidth, crossing number, or cliquewidth of the resulting graphs.

THEOREM 4.1. *For every integer $d \geq 1$, there is some $c_d \in \mathbb{N}$ such that the following holds. Let Ω be a signature graph featuring no edge weights and only Boolean signatures that are parity-consistent. Let $X \subseteq V(\Omega)$ be a set of vertices of degree at most d such that every vertex* not in *X has the signature $\mathtt{HW}_{=1}$. Then there is a $c_d \cdot n^{O(1)}$ time Turing reduction from computing* Holant(Ω) *to* #PERFMATCH *on unweighted graphs, and we can choose one of the following three statements to hold:*

1. *For every constructed instance G' of* #PERFMATCH, *we have* $\mathrm{cutw}(G') \leq \mathrm{cutw}(\Omega) + c_d$.
2. *If Ω is a graph embedded in the plane and there are t vertices in X whose signatures can be realized by planar matchgates, then every constructed instance G' of* #PERFMATCH *has crossing number at most $c_d \cdot (|X| - t)$. Here, we assume that all used planar matchgates feature only weights* 1 *and* -1.
3. *If Ω has no parallel edges and has a (k,s)-expression where every vertex of X is singular, then every constructed instance G' of* #PERFMATCH *has* $\mathrm{cw}(G') \leq k + s \cdot c_d$.

Proof. By Lemma 3.3, we obtain a matchgate Γ_v on $O(2^d d)$ vertices for each parity-consistent signature f_v at $v \in V(\Omega)$ of arity d. Let G denote the graph on edge weights $\{-1, 1/2, 1\}$ obtained from Ω by inserting Γ_v at v, for all v. By Fact 3.2, we have $\mathrm{Holant}(\Omega) = \mathrm{PerfMatch}(G)$.

To reduce to unweighted #PERFMATCH, it remains to remove the edge weights -1 and $1/2$ from G. To this end, we apply a standard interpolation argument: Introduce two indeterminates x, y and replace each edge weight -1 by x, and each edge weight $1/2$ by y. This yields a graph $G_{x,y}$ on edge weights $1, x, y$. Then

$$p(x,y) := \mathrm{PerfMatch}(G_{x,y}) \in \mathbb{Z}[x,y]$$

is a polynomial of maximum degree $\ell := |V(G)|/2$ in the indeterminates x and y. Assume we can evaluate $p(\xi)$ on all points $\xi \in A^2$ for an arbitrary set $A \subseteq \mathbb{Q}$ with $|A| = \ell + 1$. That is, we can evaluate $\mathrm{PerfMatch}(G_\xi)$ for graphs on edge weights $\{1\} \cup A$. Then we can use multivariate polynomial interpolation (see [30]) to obtain all coefficients of p in time $\ell^{O(1)}$. In particular, we can then evaluate $p(-1, 1/2) = \mathrm{PerfMatch}(G)$.

In the following, the set A will be chosen according to the parameter we wish to bound. We then evaluate $p(\xi)$ for $\xi \in A^2$ by means of certain gadgets/matchgates that simulate the edge weights from A. To this end, we present two different ways of choosing A and simulating the edge weights (see Figure 5).

Method 1. We let $A = \{2^i \mid i \in [\ell]\}$ and observe that an edge ab of weight 2^i can be simulated with a path of length $2i - 1$ on edges $e_1, \ldots, e_{2i-1}$, where edge $e_{2\kappa-1}$ for $\kappa \in [i]$ has weight 2. Then an edge of weight 2 between vertices u and v can in turn be simulated by two parallel edges between u and v, each of which is subdivided twice. We obtain a gadget of cutwidth 2 on $O(\ell)$ vertices.

Method 2. We let $A = [\ell]$ and use a construction from [37] to simulate the weight $i \in [\ell]$ with $O(\log^2 i)$ vertices and pathwidth 2. It is clear that we can simulate an edge of weight $r + s$ by two parallel edges of weights r and s, respectively. Hence, we can simulate weight $i \in [\ell]$ by at most $\lceil \log i \rceil$ parallel edges by introducing an edge for each 1-bit in the binary representation of i: if the κ-th bit is 1, then we introduce a parallel edge of weight $2^{\kappa-1}$. Then we realize each weight $2^{\kappa-1}$ by a path on $O(\kappa)$ vertices as in Method 1.

Using interpolation as described above, we can then reduce the computation of $\mathrm{Holant}(\Omega)$ to unweighted #PERFMATCH with either of the two methods. To ensure that one of the three chosen statements holds, we choose between the two methods in the following way:

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

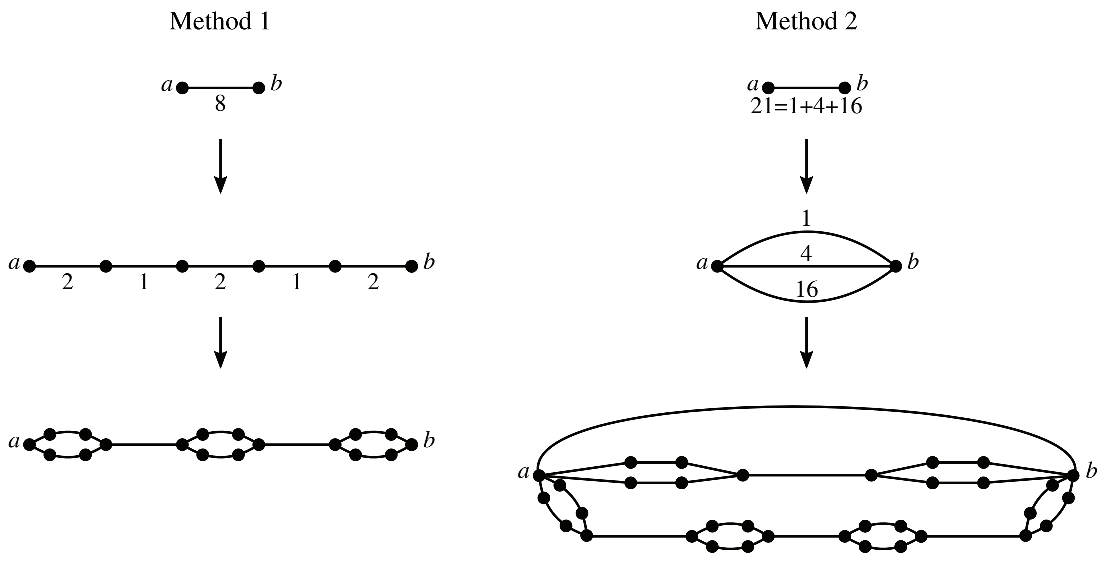


Figure 5: Removing weights using Method 1 and Method 2 in the proof of Theorem 4.1.

In the case of **cutwidth**, we use Method 1. Every weighted matchgate Γ to be inserted into Ω has size bounded by a function of d, hence its cutwidth is also bounded by a function of d. Method 1 can be expressed as subdividing edges of Γ a certain number of times, replacing some edges with two parallel edges, and then again subdividing some edges. Subdividing edges does not increase cutwidth and duplicating edges increases cutwidth at most by a factor of 2, hence the matchgates have cutwidth bounded by a function of d even after simulating the weights. Then Lemma 2.3 shows that inserting these gadgets increases cutwidth at most by a function of d.

In the case of **crossing number**, we use Method 1. We can extend the planar drawing of Ω to a drawing of G that features crossings only in the drawings of nonplanar matchgates Γ. Subdividing edges does not increase the crossing number, duplicating edges increases it at most by a factor of 4. Thus each nonplanar matchgate has crossing number bounded by some function of d, even after simulation of weights. Since Ω features at most $(|X|-t)$ nonplanar matchgates, we obtain that G has crossing number at most $(|X|-t)\cdot c_d$ for some suitably large constant c_d. Note that no crossings are introduced into planar matchgates by simulating edge-weights.

In the case of **cliquewidth**, we use Method 2. Every weighted matchgate Γ has size bounded by a function of d, hence its pathwidth is also bounded by a function of d. Method 2 can be expressed as replacing edges with several parallel copies, subdividing edges several times, duplicating certain edges, and then again subdividing them. Replacing an edge with parallel edges has no effect on pathwidth and subdividing a subset of the edges (even multiple times) can increase pathwidth at most by 1, as shown in [5]. Thus, after the application of Method 2, the unweighted gadgets still have pathwidth bounded by a function of d. Then by Lemma 2.4, the graph G' obtained after the insertion of the weighted gadgets has cliquewidth at most $k+d(s+1)$ plus a constant depending only on d, hence we can indeed bound $\mathrm{cw}(G')$ by $k+s\cdot c_d$ for a suitably large constant c_d. This concludes the proof.

5 Parameterizing by cutwidth or crossing number

In this section, we prove Theorems 1.2 and 1.7 by a lower bound for parameterization by cutwidth (implying the lower bound for parameterization by pathwidth and treewidth) and by a lower bound for parameterization by crossing number (implying the lower bound for parameterization by genus).

We first show how to reduce counting the number of satisfying assignments of an n-variable m-clause d-CNF formula φ to computing the Holant of certain signature graphs. Then we invoke Theorem 4.1 to reduce the resulting Holant problem to #PERFMATCH on graphs with $O(nm)$ vertices and edges that have cutwidth $n+O(1)$ or crossing number $O(n+dm)$.

5.1 Parameterizing by cutwidth.

Let φ be a d-CNF formula on n variables $x_1,\dots,x_n$ and m clauses $C_1,\dots,C_m$. As an intermediate step in the reduction to #PERFMATCH, we derive two signature graphs from φ, following the structure outlined in Figure 6.

That is, each graph consists essentially of an $n\times m$ square grid whose rows and columns correspond to the variables and clauses of φ, respectively. The horizontal

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

edges in row $i \in [n]$ will serve to propagate a binary assignment to the variable x_i, and in column $j \in [m]$, we will test whether the assignment to $x_1, \dots, x_n$ encoded this way satisfies clause C_j.

To realize this construction, we need to define certain auxiliary Boolean signatures that play the role of negative and positive (and neutral) literals in φ.

DEFINITION 6. *We define the Boolean 6-ary signatures* $\mathtt{LIT}_0$*,* $\mathtt{LIT}_-$ *and* $\mathtt{LIT}_+$ *on the following six inputs, grouped into three pairs of inputs*

$$(x_{in}, x_{out}),\ (x_{top1}, x_{top2}),\ (x_{btm1}, x_{btm2}).$$

For any assignment $a \in \{0,1\}^6$, the signatures yield value 1 iff all of the following holds: Firstly, for each input pair, the two values in the pair agree. That is, we have $a(x_{in}) = a(x_{out})$ and $a(x_{top1}) = a(x_{top2})$ and $a(x_{btm1}) = a(x_{btm2})$.

Secondly, if the top input pair is active, then so is the bottom input pair, regardless of the assignment to any other inputs. That is, if $a(x_{top1}) = a(x_{top2}) = 1$, then $a(x_{btm1}) = a(x_{btm2}) = 1$.

On the other hand, if the top input pair is inactive, that is, $a(x_{top1}) = a(x_{top1}) = 0$, then $\mathtt{LIT}_0$*,* $\mathtt{LIT}_-$ *and* $\mathtt{LIT}_+$ *differ in their behavior:*

- *For* $\mathtt{LIT}_0$*, we require $a(x_{btm1}) = a(x_{btm2}) = 0$.*
- *For* $\mathtt{LIT}_+$*, require $a(x_{btm1}) = a(x_{btm2}) = a(x_{in})$.*
- *For* $\mathtt{LIT}_-$*, require $a(x_{btm1}) = a(x_{btm2}) = a(\neg x_{in})$.*

It is evident that $\mathtt{LIT}_0$, $\mathtt{LIT}_-$ and $\mathtt{LIT}_+$ are all even signatures. As a matter of fact, the inputs x_{top1}, x_{top2} and x_{btm1}, x_{btm2} are defined to come in pairs with enforced equality for no other reason than to ensure this.

All signatures defined above propagate the assignment of x_{in} to x_{out}. This ensures that, in every satisfying assignment to the constructed signature graph, the same binary value will be assigned to all horizontal edges. If $x_{top1} = 1$, then we require $x_{btm1} = 1$ as well. This ensures that if a clause is satisfied by previous literals, then this information will be propagated to the next literal. If $x_{top1} = 0$, then $\mathtt{LIT}_0$ simply passes this information to x_{btm1}. The signatures $\mathtt{LIT}_-$ and $\mathtt{LIT}_+$ however check whether x_{in} is assigned 1 (in which case a positive literal would be satisfied and $\mathtt{LIT}_+$ assigns x_{in} to x_{btm1}) or 0 (in which case $\mathtt{LIT}_-$ assigns $\neg x_{in}$ to x_{btm1}). We can now define the relevant signature graphs from φ.

LEMMA 5.1. *Let φ be a d-CNF formula on n variables and m clauses. Then we can construct two unweighted signature graphs Ω_0 and Ω_1 such that*

$$\#\mathrm{SAT}(\varphi) = \mathrm{Holant}(\Omega_0) + \mathrm{Holant}(\Omega_1). \tag{5.6}$$

Furthermore, both Ω_0 and Ω_1 are planar, have cutwidth $n + O(1)$ and maximum degree $O(1)$.

Proof. We define the graphs underlying Ω_0 and Ω_1 as follows, see also Figure 6:

1. Create an $n \times m$ grid of vertices $v_{i,j}$ for $i \in [n]$ and $j \in [m]$. Copy each vertical edge, such that $v_{i,j}$ and $v_{i+1,j}$ are connected by two parallel edges for $i \in [n-1]$ and $j \in [m]$. We will later describe how to assign signatures to these vertices.
2. Create two additional sets of vertices $\{v_{i,0} \mid i \in [n]\}$ and $\{v_{i,m+1} \mid i \in [n]\}$. For each $i \le n-1$, connect $v_{i,0}$ to $v_{i+1,0}$ and to $v_{i,1}$. Connect $v_{n,0}$ to $v_{n,1}$. Also connect $v_{i,m+1}$ to $v_{i+1,m+1}$ and to $v_{i,m}$ for each $i \le n-1$. Connect $v_{n,m+1}$ to $v_{n,m}$.
3. In the case of Ω_0, assign the signature $\mathtt{EVEN}$ of appropriate arity to each vertex created in the previous step. Define Ω_1 likewise, but assign $\mathtt{ODD}$ to the two vertices $v_{n,0}$ and $v_{n,m+1}$. Note that the vertex sets $\{v_{i,0} \mid i \in [n]\}$ and $\{v_{i,m+1} \mid i \in [n]\}$ each effectively express an $\mathtt{EVEN}$ gate of arity n in Ω_0, and an $\mathtt{ODD}$ gate of arity n in Ω_1. That is, we may contract, say, $\{v_{i,0} \mid i \in [n]\}$ to a single vertex of arity n with signature $\mathtt{EVEN}$ without changing the Holant of Ω_0. However, in order to use Theorem 4.1, we would like to use only $\mathtt{EVEN}$ and $\mathtt{ODD}$ gates of constant arity.
4. Create m additional vertices $v_{n+1,j}$ for $j \in [m]$. For all $j \in [m]$, connect $v_{n+1,j}$ to $v_{n,j}$ with two parallel edges and assign $\mathtt{HW}_{=2}$ to $v_{n,j}$.
5. Create m additional vertices $v_{0,j}$ for $j \in [m]$. For all $j \in [m]$, connect $v_{0,j}$ to $v_{1,j}$ with two parallel edges and assign $\mathtt{HW}_{=0}$ to $v_{0,j}$.

It is evident from the construction and the drawing in Figure 6 that Ω_0 and Ω_1 are planar. Furthermore, if we enumerate the vertices of the graphs column by column, we obtain linear layouts with cutwidth at most $n + c$ for a constant $c \in \mathbb{N}$ independent of the input.

It remains to assign signatures to the vertices $v_{i,j}$ for $i \in [n]$ and $j \in [m]$ constructed in the first step. Recall that φ has variables $x_1, \dots, x_n$ and clauses $C_1, \dots, C_m$. We define an array $A \in \{0,+,-\}^{n \times m}$ by

$$A(i,j) = \begin{cases} 0 & x_i \text{ does not appear in } C_j \\ + & x_i \text{ appears in } C_j \\ - & \neg x_i \text{ appears in } C_j \end{cases}$$

and assign to $v_{i,j}$ the signature $\mathtt{LIT}_{A(i,j)}$. In the following, we verify that $\mathrm{Holant}(\Omega_0)$ indeed counts the satisfying assignments $a : \{x_1, \dots, x_n\} \to \{0,1\}$ for φ with even Hamming weight. The same can be verified for Ω_1 and the assignments of odd weight. As a notational simplification, for $i \in [n]$, we write $v_{i,\star}$ for the vertices in row i, and for $j \in [m]$, we write $v_{\star,j}$ for the vertices in column j.

For each $i \in [n]$, the signatures $\mathtt{LIT}_0$, $\mathtt{LIT}_-$ and $\mathtt{LIT}_+$ in $v_{i,\star}$ ensure that all horizontal edges between vertices in $v_{i,\star}$ have the same value a_i. By the two columns of $\mathtt{EVEN}$

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

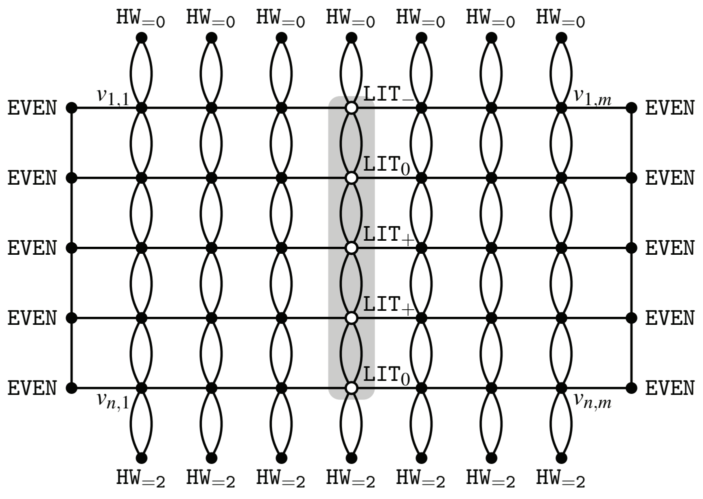


Figure 6: A drawing of Ω_0 with $n = 5$ and $m = 7$. As an example, the column corresponding to clause $C_4 = (\bar{x}_1 \vee x_3 \vee x_4)$ is highlighted.

signatures (which act like two vertices of degree n with signature EVEN), evenly many rows have active edges. Altogether, this implies that the satisfying assignments to Ω_0 encode even assignments $a : \{x_1, \dots, x_n\} \to \{0,1\}$ to the formula φ.

For each $j \in [m]$, we propagate along the vertical edges in column $v_{\star,j}$ whether the clause C_j is satisfied by the assignment a. For each $1 \leq i \leq n$, the vertex $v_{i,j}$ has two top edges whose assignment encodes whether C_j is satisfied by $a_1, \dots, a_i$. At $v_{1,j}$, this value is false, as ensured by the HW$_{=0}$ signatures at $v_{0,j}$. If the variable x_j does not appear in the clause C_j, then we propagate the assignment at the top edges downward by definition of LIT$_0$. If x_j appears positively or negatively, then we check whether x_i satisfies C_j and propagate the result downward by definition of LIT$_-$ or LIT$_+$. At the bottom of each column $j \in [m]$, the vertex $v_{n+1,j}$ of signature HW$_{=2}$ tests whether clause C_j was satisfied by the assignment $a = a_1 \dots a_n$.

Altogether, this shows that the satisfying assignments to Ω_0 encode the satisfying assignments to φ where an even number of variables is set to true. A similar proof applies for Ω_1. This proves (5.6) and hence the theorem.

By invoking Theorem 4.1, we obtain hardness for the parameterization by cutwidth.

Proof. of Theorem 1.2. Given a CNF formula φ on n variables, invoke Lemma 5.1 to obtain signature graphs Ω_0 and Ω_1 of cutwidth $n + O(1)$ that satisfy (5.6). Using Theorem 4.1 and choosing statement 1, we can then determine Holant(Ω_i) for $i \in \{0,1\}$ with an oracle for unweighted #PERFMATCH that asks only queries on graphs with $n^{O(1)}$ vertices and cutwidth $n + O(1)$. Thus an algorithm with running time $(2-\varepsilon)^{\mathrm{cutw}(G)} n^{O(1)}$ for some $\varepsilon > 0$ would yield a $(2-\varepsilon)^n m^{O(1)}$ time algorithm for n-variable m-clause CNF-SAT, violating #SETH. As the treewidth of a graph is always bounded from above by its cutwidth, the theorem follows.

5.2 Parameterizing by crossing number. With a slight modification, the construction of Lemma 5.1 also allows us to prove the lower bound for #PERFMATCH parameterized by crossing number. To this end, we first need to observe that the signature LIT$_0$ can *almost* be realized by a planar matchgate. That is, we can realize a signature LIT$_0{}^*$ that agrees with LIT$_0$ on assignments $a \in \{0,1\}^6$ that give the same value to x_{top1} and x_{top2}, but LIT$_0{}^*$ may attain arbitrary values on all other assignments.

LEMMA 5.2. *There is a signature* LIT$_0{}^*$ *that can be realized by a planar matchgate (on edge-weights* 1 *and* -1*) and satisfies, on all* $a \in \{0,1\}^6$ *with* $a(x_{top1}) = a(x_{top2})$*, the condition* LIT$_0(a) =$ LIT$_0{}^*(a)$*. The signature* LIT$_0{}^*$ *may however attain arbitrary values on all other* $a \in \{0,1\}^6$.

From this, we can conclude the desired lower bound.

Proof. of Theorem 1.7. Since we wish to prove a lower bound under #ETH, we may reduce from #SAT on 3-CNF formulas. Given such a formula φ on n variables and m clauses, it is clear that φ contains at most $3m$ literals that occur positively or negatively.

We construct the planar signature graphs Ω_0 and Ω_1 from φ described in Lemma 5.1, replacing each occurrence of LIT$_0$ by LIT$_0{}^*$. It can be checked that this particular replacement preserves Holants: At the top-most vertex $v_{1,j}$ of each column $j \in [m]$, the two inputs x_{top1} and x_{top2} are forced to the same assignment, and by definition of LIT$_0{}^*$, LIT$_-$ and LIT$_+$, the same applies inductively

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

at all vertices $v_{i,j}$ for $i \in [n]$. Hence, under any satisfying assignment, $\mathtt{LIT}_0$ would yield the same value as $\mathtt{LIT}_0{}^*$.

As described above, there are at most $3m$ occurrences of the non-planar signatures $\mathtt{LIT}_-$ and $\mathtt{LIT}_+$ in Ω_0 and Ω_1. In addition, there are $O(n+m)$ occurrences of the signatures $\mathtt{HW}_{=2}$, ODD, and EVEN, which in fact have planar matchgates, but we do not use this here. Thus, using Theorem 4.1, we obtain a reduction from #SAT on 3-CNF formulas with n variables and m clauses to #PERFMATCH with $O(nm)$ vertices and crossing number $O(n+m)$. Consequently, an algorithm for #PERFMATCH with running time $2^{o(k)}n^{O(1)}$ on graphs with crossing number k would imply a $2^{o(n+m)}n^{O(1)}$ time algorithm for #SAT on 3-CNF formulas, and thus refute #ETH by Lemma 2.2.

6 Parameterizing by cliquewidth

In this section, we prove the following conditional lower bound under #SETH that complements Theorem 1.3. This lower bound then implies Theorem 1.4.

THEOREM 6.1. *For any fixed $k \in \mathbb{N}$ and $\varepsilon > 0$, the following holds: If we can solve* #PERFMATCH *in time $\mathcal{O}(|T| \cdot n^{k-2-\varepsilon})$ on an unweighted graph G given with an expression for G with k large and $\mathcal{O}(1)$ singleton labels, then* #SETH *fails.*

Since expressions with k large and $\mathcal{O}(1)$ singleton labels trivially have $k+\mathcal{O}(1)$ labels, this indeed proves Theorem 1.4. We note that it is crucial to obtain unweighted graphs, as the evaluation of PerfMatch is trivially #P-hard on edge-weighted cliques, whose underlying graphs have cliquewidth 2. (We can simply assign weight zero to non-edges. If zero-weight edges are explicitly forbidden, then a simple interpolation argument allows to simulate them by edges of non-zero weight.)

6.1 The colored hitting set problem. To prove Theorem 6.1, we reduce from counting colorful hitting k-sets. This is a simple variant of the hitting k-set problem, for which a tight lower bound was already shown by Patrascu and Williams [75].

DEFINITION 7. *In the problem* #COLHS*, the input consists of k disjoint universes $X = (X_1, \dots, X_k)$ with n individuals each and sets $A = (A_1, \dots, A_m)$ with $A_j \subseteq X_1 \cup \dots \cup X_k$ for $j \in [m]$. We may assume that $X_i = \{i\} \times [n]$ for all $i \in [k]$. The task is then to determine the number of colorful hitting sets, that is, the size of the set*

$$\mathrm{COLHS}(X,A,k) = \{(s_1,\dots,s_k) \in X_1 \times \dots \times X_k \mid \\ \forall j \in [m] : \exists i \in [k] : s_i \in A_j\}.$$

The difference to the usual hitting k-set problem is that the universe X can be considered to be colored by k colors, and we are looking for a colorful hitting set. A tight lower bound for this problem can be established in a similar way as in [75].

LEMMA 6.1. *For each fixed $k \geq 2$, the following holds: If there exists an $\varepsilon > 0$ such that, for each $d \in \mathbb{N}$, there is an $O(n^{k-\varepsilon})$ time algorithm for* #COLHS *on instances with universe size n and $O(k^d \cdot \log^d n)$ sets, then* #SETH *fails. Here, the constant factor in the running time may depend on d.*

6.2 Construction of the signature graph. In the remainder of this section, we show how to solve an instance (X,A,k) for the problem #COLHS by reduction to the problem #PERFMATCH on unweighted graphs of cliquewidth $k + O(1)$. To this end, we first construct a grid-like signature graph Ω on k rows and m columns containing *cell* gates $C_{i,j}$ for $i \in [k]$ and $j \in [m]$. The i-th row corresponds to the individual from X_i to be picked in our hitting set, while the j-th column, which we denote by D_j, corresponds to the set A_j.

The construction resembles that of Section 5: Each row i propagates an element $x_i \in X_i$ from left to right, and each column D_j will check whether A_j is hit by the elements $\{x_1, \dots, x_k\}$ propagated by the rows. The checks at each column D_j are performed from top to bottom, so that cell $C_{i,j}$ will know the value of x_i and whether A_j was already hit by $\{x_1, \dots, x_{i-1}\}$. This allows $C_{i,j}$ to determine whether A_j is hit by $\{x_1, \dots, x_i\}$ and to propagate this information to the cell $C_{i+1,j}$ below. When reaching the bottom-most cell $C_{k,j}$, the column D_j will know whether A_j is hit by $\{x_1, \dots, x_k\}$. The main difference to Section 5 is that we do not transfer a Boolean value along the i-th row, but rather a *number* $x_i \in [n]$, which encodes the element (i, x_i) chosen in our hitting set. In particular, cell gates will therefore feature $O(n)$ dangling edges (rather than a constant number).

In the following, we give a formal construction of a cell by first specifying its underlying graph. We will then attach signatures to this graph to obtain a gate.

DEFINITION 8. *A cell is a gate containing pairs of dangling edges (e_{top1}, e_{top2}) and (e_{btm1}, e_{btm1}) and (a_κ, b_κ) for $\kappa \in [n]$. It is defined as follows; see also Figure 7.*

1. *Create vertices $c_1, \dots, c_n$. For all $\kappa \in [n]$, connect c_κ to the dangling edges a_κ and b_κ.*
2. *For $1 \leq \kappa \leq n-1$, connect c_κ to $c_{\kappa+1}$ by 4 parallel edges. Add (e_{top1}, e_{top2}) as dangling edges to c_1 and add (e_{btm1}, e_{btm2}) to c_n.*
3. *Connect c_1 to an extra vertex u_{top} of signature $\mathtt{HW}_{=2}$ with 2 parallel edges, and connect c_n to an extra vertex u_{btm} of signature $\mathtt{HW}_{=0}$ with 2 parallel edges.*

Intuitively speaking, the 4 parallel edges between consecutive vertices in a cell are used to transfer 2 bits in a

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

parity-consistent way. We require the cell $C_{i,j}$ to behave as follows. For every satisfying assignment $x \in \{0,1\}^{E(C_{i,j})}$ to $C_{i,j}$, the following conditions have to be fulfilled:

Doubling: *For each pair of dangling edges, its two edges must agree in their values on x. This implies in particular that the assignment to $a_1,\dots,a_n$ is propagated to $b_1,\dots,b_n$.*

Block: *There exists a number $t \in [n]$ such that all dangling edges a_κ, b_κ for $1 \le \kappa \le t$ are active in x, and all dangling edges a_κ, b_κ for $\kappa > t$ are inactive.*
That is, the active dangling edges in any satisfying assignment appear in a consecutive block starting at a_1 and b_1. The length of this block will encode the propagated element from $[n]$.

Propagation: *If (e_{top1}, e_{top2}) is active, then so is (e_{btm1}, e_{btm2}). This will later ensure that, if a set A_j was already hit by elements $x_1,\dots,x_{i-1}$ preceding x_i, then this information is propagated down. If (e_{top1}, e_{top2}) is inactive, then (e_{btm1}, e_{btm2}) is active iff the set A_j is hit by (i,t), where t is the number defined in the block condition.*

It remains to add signatures to the cell $C_{i,j}$ to ensure these conditions. To this end, we define the following "master" signatures $\texttt{MSTR}_0$ and $\texttt{MSTR}_+$ on five pairs of dangling edges. See also Figure 7. There is one "wire" pair, two pairs that we denote as "Boolean" in-/outputs, and two pairs that we denote as "thru" in-/outputs.

These signatures are used in a similar way as the signatures $\texttt{LIT}_0$ and $\texttt{LIT}_+$ from Section 5. If $i \in [k]$, $j \in [m]$ and $\kappa \in [n]$ are such that $(i,\kappa) \in A_j$, then we place $\texttt{MSTR}_+$ at the vertex c_κ in the cell $C_{i,j}$. Otherwise, we place $\texttt{MSTR}_0$ at c_κ.

The main difference to Section 5 is the presence of dangling edges that are considered as "thru" in-/outputs. These will ensure that active dangling edges of the cell $C_{i,j}$ appear in a consecutive block as defined above: An inactive thru input at c_κ enforces that the thru output of c_κ is inactive as well, and we use this to ensure that the active edges among $a_1,\dots,a_n$ appear as a consecutive block. Due to the vertex u_{top}, the thru inputs at c_1 are forced to be active.

The Boolean in-/outputs of $\texttt{MSTR}_0$ and $\texttt{MSTR}_+$ at c_κ will then be used to propagate whether the set A_j was hit by any of the elements $(i,1),\dots(i,\kappa)$, precisely as for the signatures $\texttt{LIT}_0$ and $\texttt{LIT}_+$ from Section 5: In the case of $\texttt{MSTR}_0$ (which we use if $(i,\kappa) \notin A_j$), we simply propagate the Boolean inputs to the outputs, as seen for $\texttt{LIT}_0$. In the case of $\texttt{MSTR}_+$ (which we use if $(i,\kappa) \in A_j$), we test whether a_κ is the first inactive edge after the block of active edges among $a_1 \dots, a_n$. If this is the case, then A_j is hit by (i,κ), and we propagate this information down to the Boolean output pair. This is parallel to the behaviour of $\texttt{LIT}_+$.

In the following, we give formal definitions for the signatures $\texttt{MSTR}_0$ and $\texttt{MSTR}_+$.

DEFINITION 9. *The master signatures $\texttt{MSTR}_0$ and $\texttt{MSTR}_+$ are 10-ary Boolean signatures*

$$\texttt{MSTR}_0, \texttt{MSTR}_+ : \{0,1\}^{10} \to \{0,1\}$$

with inputs

$$(x^{in}, x^{out}),\ (e^{in}_{bool1}, e^{in}_{bool2}),\ (e^{out}_{bool1}, e^{out}_{bool2}),\ (e^{in}_{thru1}, e^{in}_{thru2}),\ (e^{out}_{thru1}, e^{out}_{thru2}).$$

They are defined such that $\texttt{MSTR}_0(x) = 1$ and $\texttt{MSTR}_+(x) = 1$ hold for $x \in \{0,1\}^{10}$ iff all of the following conditions are fulfilled:

1. *For each pair of dangling edges, such as (x^{in}, x^{out}), the two dangling edges in x have the same assignment. In the remainder of this definition, we will therefore refer only to one of the two dangling edges, say to x^{in}.*
2. *If e^{in}_{thru1} is inactive, then so is e^{out}_{thru1}.*
   If e^{in}_{thru1} is active, then e^{out}_{thru1} is active iff x^{in} is.
3. *If e^{in}_{bool1} is active, then so is e^{out}_{bool1}. Furthermore:*
   (a) *In the case of $\texttt{MSTR}_0$, if e^{in}_{bool1} is inactive, then e^{out}_{bool1} is inactive as well.*
   (b) *In the case of $\texttt{MSTR}_+$, if e^{in}_{bool1} is inactive, then e^{out}_{bool1} is active iff e^{in}_{thru1} is active and x^{in} is inactive*

We apply these signatures as follows to the vertices of a cell $C_{i,j}$: For all $\kappa \in [n]$, if $(i,\kappa) \in A_j$, we place $\texttt{MSTR}_+$ at c_κ. If $(i,\kappa) \notin A_j$, then we place $\texttt{MSTR}_0$ at c_κ. This concludes the definition of the cell $C_{i,j}$.

It is clear from the definitions of $\texttt{MSTR}_0$ and $\texttt{MSTR}_+$ that both signatures are even. Furthermore, we can verify that the cell $C_{i,j}$ satisfies the conditions of Definition 8 (doubling, block, propagation): The doubling condition follows from item 1 of Definition 9, the block condition follows from item 2, and the propagation condition follows from item 3.

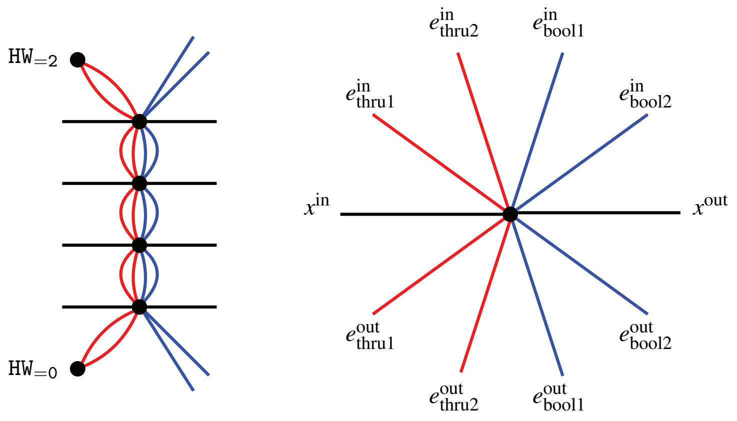
To proceed, for any $j \in [m]$, we connect the cells $C_{i,j}$ for $i \in [n]$ into the *column D_j*.

DEFINITION 10. *For any $j \in [m]$, we define the* column *D_j as follows: First, we create a disjoint union of cells $C_{1,j},\dots,C_{k,j}$. Denote the subset of dangling edges $\{a_\kappa \mid \kappa \in [n]\}$ in $C_{i,j}$ by $F^{in}_{i,j}$, and the subset $\{b_\kappa \mid \kappa \in [n]\}$ by $F^{out}_{i,j}$. This results in $2k$ sets $F^{in}_{i,j}$ and $F^{out}_{i,j}$ for $i \in [k]$.*

Within a column, we then connect cells in the following way:

1. *For $1 \le i \le n-1$, connect the input pair (e_{btm1}, e_{btm2}) of $C_{i,j}$ with the input pair (e_{top1}, e_{top2}) of $C_{i+1,j}$ to obtain two parallel edges.*

**Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.**



$\mathtt{HW}_{=2}$
$\mathtt{HW}_{=0}$
$e^{\text{in}}_{\text{thru2}}$
$e^{\text{in}}_{\text{bool1}}$
$e^{\text{in}}_{\text{thru1}}$
$e^{\text{in}}_{\text{bool2}}$
x^{in}
x^{out}
$e^{\text{out}}_{\text{thru1}}$
$e^{\text{out}}_{\text{bool2}}$
$e^{\text{out}}_{\text{thru2}}$
$e^{\text{out}}_{\text{bool1}}$

Figure 7: A cell and a vertex of signature MSTR.

2. *Add two extra vertices u_0 and u_{n+1}. Assign* $\mathtt{HW}_{=0}$ *to u_0 and* $\mathtt{HW}_{=2}$ *to u_{n+1}, similarly as in Lemma 5.1. Connect u_0 with two parallel edges to the input pair (e_{top1}, e_{top2}) of $C_{1,j}$, and likewise connect u_{n+1} to the input pair (e_{btm1}, e_{btm2}) of $C_{n,j}$.*

Note that the resulting gate D_j has $2kn$ dangling edges, grouped into the $2k$ sets $F^{in}_{i,j}$ and $F^{out}_{i,j}$ for $i \in [k]$, each of which has size n.

We observe that the column D_j has a Boolean signature when considered as a gate. For $i \in [k]$, the propagation condition of cell $C_{i,j}$ ensures that the assignments for $F^{in}_{i,j}$ and $F^{out}_{i,j}$ agree. Furthermore, by construction of the cells $C_{i,j}$ for $i \in [k]$, if x is a satisfying assignment to D_j, then there are numbers $t_1, \ldots, t_k \in [n]$ such that precisely the first t_i edges in $F^{in}_{i,j}$ are active, while no other edges are active. Finally, we can verify that the column D_j yields value 1 on an input x as above iff the set $\{(i, t_i) \mid i \in [k]\}$ hits the set A_j.

To conclude our construction, we connect the m columns $D_1, \ldots, D_m$ to a signature graph Ω. This requires a slightly cumbersome construction to ensure that each colorful hitting k-set in (X, A, k) corresponds to a fixed number of satisfying assignments in G while preserving a low cliquewidth. To this end, we "wrap" the cells of columns by bicliques, as demonstrated in Figure 8 and specified in the following.

DEFINITION 11. *Let (X, A, k) be an instance to #COLHS with universes $\{i\} \times [n]$ for $i \in [k]$ and sets $A_1, \ldots, A_m$. Without limitation of generality, we may assume that each A_j for $j \in [m]$ contains only elements (i, κ) with even $\kappa \in [n]$. Proceed as follows to obtain $\Omega = \Omega(X, A, k)$:*

1. *For $j \in [m]$, construct the column D_j as in Definition 10. Let $F^{in}_{i,j}$ and $F^{out}_{i,j}$ for $i \in [k]$ denote its sets of dangling edges.*
2. *For each $i \in [k]$ and $j \in [m]$, add four independent sets $I^{(t)}_{i,j}$ of size n, for $t \in [4]$, and denote its vertices by $I^{(t)}_{i,j} = \{a^{(t)}_{i,j,1}, \ldots, a^{(t)}_{i,j,n}\}$. Assign the signature* $\mathtt{HW}_{=1}$ *to these vertices.*
3. *For $\kappa \in [n]$, connect vertex c_κ to $a^{(1)}_{i,j,\kappa}$ and to $a^{(2)}_{i,j,\kappa}$. Then connect $a^{(2)}_{i,j,\kappa}$ to $a^{(3)}_{i,j,\kappa}$.*
4. *Add all edges between vertices in $I^{(3)}_j$ and $I^{(4)}_j$ to obtain a complete bipartite graph between these sets. Consider Figure 8 for an example.*
5. *For $1 \le j \le m-1$, add all edges between vertices in $I^{(4)}_j$ and $I^{(1)}_{j+1}$, see also Figure 8.*
6. *Add two vertices u^* and v^*.*
   (a) *Delete the independent set $I^{(1)}_1$, so that the edges in $F^{in}_{1,1} \cup \ldots \cup F^{in}_{n,1}$ of the left-most column become dangling edges. Connect them to u^* and assign the signature* EVEN *to u^*.*
   (b) *Delete the independent sets $I^{(2)}_m$, $I^{(3)}_m$, $I^{(4)}_m$, so that the edges in $F^{out}_{1,m} \cup \ldots \cup F^{out}_{n,m}$ of the right-most column become dangling edges. Connect them to v^* and assign the signature* EVEN *to v^*.*
   (c) *Finally, as in Section 3, replace u^* and v^* by paths of vertices on* EVEN *signatures of arity 3.*

FACT 6.1. *For an instance to #COLHS with k universes of size n each, and m sets, the signature graph Ω constructed above has $O(kmn)$ vertices. Furthermore, all signatures other than* $\mathtt{HW}_{=1}$ *in it have constant arity.*

6.3 Correctness of the signature graph.

We prove that, for $\Omega = \Omega(X, A, k)$, the quantity $\text{Holant}(\Omega)$ counts colorful hitting sets in (X, A, k) up to a simple factor.

LEMMA 6.2. *Let (X, A, k) be an instance to #COLHS with $X_i = \{i\} \times [n]$ for $i \in [k]$ and sets $A_1, \ldots, A_m$. Let $\Omega = \Omega(X, A)$ be constructed as in Definition 11. Then*

$$\text{Holant}(\Omega) = (n!)^{k(m-1)} \cdot \#\text{COLHS}(X, A, k).$$

Proof. Consider the satisfying assignments for Ω. It follows from Definition 8 that, for each $i \in [k]$, there exists

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

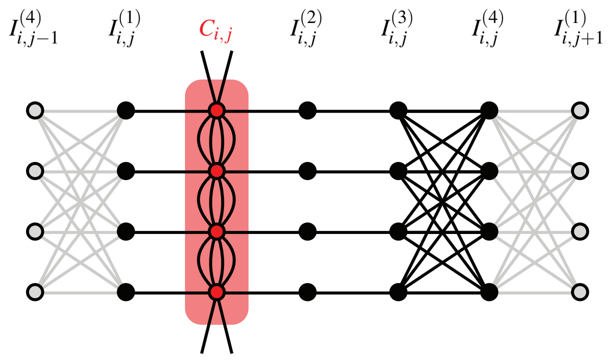


Figure 8: The subgraph of G containing the cell $C_{i,j}$ and its surrounding independent sets

a number $x_i \in [n]$ such that the first x_i dangling edges in $F_{i,1}^{in}$ and $F_{i,1}^{out}$ are active, while no other edges in these sets are active. Please recall Figure 8 in the following.

- Since the first x_i dangling edges in $F_{i,1}^{in}$ and $F_{i,1}^{out}$ are active, while no other dangling edges are active, the following holds: Among the independent set $I_{i,1}^{(2)}$, precisely the first x_i vertices are matched by dangling edges of the cell $C_{i,j}$.
- This in turn implies that exactly the last $n-x_i$ edges are active between $I_{i,1}^{(2)}$ and $I_{i,1}^{(3)}$, so the last $n-x_i$ vertices of $I_{i,1}^{(3)}$ are matched by edges from $I_{i,1}^{(2)}$ to $I_{i,1}^{(3)}$.
- Consequently, there are x_i active edges between $I_{i,1}^{(3)}$ and $I_{i,1}^{(4)}$. Since their endpoints in $I_{i,1}^{(3)}$ are fixed (to the first x_i vertices, as the last $n-x_i$ vertices are already matched), there are $(n)_{x_i}$ ways of choosing these active edges. Here, we denote by $(n)_t$ the falling factorial $(n)_t = (n)(n-1)\dots(n-t+1)$.
- Finally, there are $n-x_i$ active edges between $I_{i,1}^{(4)}$ and $I_{i,2}^{(1)}$. The endpoints of these $n-x_i$ edges in $I_{i,1}^{(4)}$ are fixed by the choices in the preceding step, and by the block property of the cell $C_{i,2}$, the endpoints of these edges in $I_{i,2}^{(1)}$ are fixed to the last $n-x_i$ vertices as well. Hence, there are $(n-x_i)!$ ways of choosing the active edges between $I_{i,1}^{(4)}$ and $I_{i,2}^{(1)}$. In particular, the first x_i edges among $F_{i,2}^{in}$ are now active. That is, the value x_i was propagated correctly from $C_{i,1}$ to $C_{i,2}$.

From this initial step, we obtain inductively that, for satisfying assignment and every $i \in [k]$ and $j \in [m]$, the first x_i edges in every set $F_{i,j}^{in}$ and $F_{i,j}^{out}$ are active. We can thus associate with each satisfying assignment x to Ω the encoding of a set $S_x = \{(i,x_i) \mid i \in [k]\}$ with $|S_x \cap X_i| = 1$ for all $i \in [k]$.

For a given set S as above, either there is no satisfying assignment x to Ω with $S = S_x$ (this occurs when at least one of the column signatures D_j yields value 0, which in turn occurs precisely if there is some set A_j that was not hit by S), or there are precisely

$$\prod_{i\in[k]} ((n)_{x_i} \cdot (n-x_i)!)^{m-1} = (n!)^{k(m-1)}$$

satisfying assignments x to Ω, each having value 1 as all involved signatures are Boolean. We use here that $\sum_i x_i$ is even to ensure that an even number of dangling edges is active in each column, thus satisfying the paths of `EVEN` signatures at the first and last columns. This proves the lemma.

6.4 Bounding the cliquewidth. Finally, we bound the cliquewidth of the graph $\Omega = \Omega(X,A,k)$ obtained from an instance (X,A,k) of counting colorful hitting k-sets via Definition 11. Let us note that parallel edges (which appear in Ω) cannot be obtained by clique expressions. We can however subdivide each parallel edge twice, and this way obtain a simple graph Ω' from Ω. If all parallel edges are singular (which can be verified for Ω), then the subdivision introduces only $O(1)$ new singleton labels.

LEMMA 6.3. *Let (X,A,k) be an instance for* #COLHS *and let $\Omega = \Omega(X,A,k)$. Then we can construct in polynomial time a $(k+2,O(1))$-expression for Ω.*

Proof. (Sketch.) We construct the graph Ω column by column. For this, we use a large forget label κ^*, large labels $\kappa_1,\dots,\kappa_k$, and an additional large working label τ. Furthermore, we use $O(1)$ singleton labels.

To construct column D_j, assume that the columns $D_1,\dots,D_{j-1}$ have been constructed already, that τ is empty, and that the label κ_i for $i \in [k]$ contains precisely the elements in the independent set $I_{i,j-1}^{(4)}$ specified in Definition 11. Then we can construct D_j and its independent sets $I_{i,j}^{(b)}$ for $i \in [k]$ and $b \in [4]$ as follows. Please consider again Figure 8.

Consider $i \in [k]$ fixed for now. For $b \in [4]$, let us write $I_{i,j}^{(b)} = \{a_{i,j,\kappa}^{(b)} \mid \kappa \in [n]\}$. For each $\kappa \in [n]$, we perform the following steps:

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

1. Construct the vertices $a^{(1)}_{i,j,\kappa}$, $a^{(2)}_{i,j,\kappa}$ and $a^{(3)}_{i,j,\kappa}$, and the cell vertex c_κ of $C_{i,j}$ on singleton labels.
2. Connect $a^{(1)}_{i,j,\kappa}$ to all vertices of label κ_i by means of the join operation. This adds all edges between $a^{(1)}_{i,j,\kappa}$ and the independent set $I^{(4)}_{i,j-1}$.
3. Connect c_κ to $a^{(1)}_{i,j,\kappa}$ and $a^{(2)}_{i,j,\kappa}$, and connect $a^{(2)}_{i,j,\kappa}$ to $a^{(3)}_{i,j,\kappa}$ by means of join operations on their singleton labels.
4. Relabel $a^{(3)}_{i,j,\kappa}$ from its singleton label to τ. Relabel $a^{(1)}_{i,j,\kappa}$, $a^{(2)}_{i,j,\kappa}$ to the forget label κ^*.
5. If $\kappa > 1$, connect $c_{\kappa-1}$ to c_κ. (Here, we assume that $c_{\kappa-1}$ is contained in a singleton label.) Relabel $c_{\kappa-1}$ to the forget label κ^*.

So far, we have constructed the cell $C_{i,j}$ and the independent sets $I^{(1)}_{i,j}$, $I^{(2)}_{i,j}$ and $I^{(3)}_{i,j}$. Our expression contains precisely the independent set $I^{(3)}_{i,j}$ in label τ. To add the independent set $I^{(4)}_{i,j}$, we proceed as follows:

1. Relabel κ_i to the forget label κ^*. Note that κ_i is empty now.
2. Add the independent set $I^{(4)}_{i,j}$ on label κ_i.
3. Join labels τ and κ_i so as to add all edges between $I^{(3)}_{i,j}$ and $I^{(4)}_{i,j}$.
4. Relabel τ to the forget label κ^*.

This finishes the construction of a single cell $C_{i,j}$ and its independent sets $I^{(b)}_{i,j}$ for $i \in [k]$ and $b \in [4]$. Repeat this process for all $i \in [k]$, and connect vertically adjacent cells $C_{i,j}$ and $C_{i+1,j}$ to columns by adding edges between vertices on singleton labels.

6.5 Finishing the proof. By combining Lemmas 6.1, 6.2 and 6.3, we obtain the proof of Theorem 6.1.

Proof. By Lemma 6.1, an algorithm with running time $O(n^{k-\varepsilon})$ for #ColHS on instances (X, A, k) with k universes of size n and $m = O(\log^d(n))$ sets for fixed $d \in \mathbb{N}$ would refute #SETH. Using Definition 11, we can transform (X, A, k) to a signature graph Ω such that

1. the value Holant(Ω) allows to recover the solution #ColHS(X, A, k) by Lemma 6.2,
2. all signatures appearing in Ω other than $\mathtt{HW}_{=1}$ are even by Fact 6.1,
3. Ω has $O(nmk) = O(n \log^d n)$ vertices by Fact 6.1,
4. and there is an expression for Ω with at most $k+2$ large labels and $O(1)$ singleton labels by Lemma 6.3. In particular, all vertices of signatures other than $\mathtt{HW}_{=1}$ are singular in this expression.

Together with Theorem 4.1, we then obtain a reduction from Holant(Ω) to #PerfMatch on graphs G' that admit clique expressions with $k+2$ large and $O(1)$ labels. The theorem also asserts that the size of these graphs is bounded by $O(n \log^{d+2} n)$. Hence, if there were an algorithm with running time $O(t^{k-2-\varepsilon})$ for solving #PerfMatch on t-vertex graphs G' that are given together with a $(k, O(1))$-expression, then we would obtain an $O((n \log^{d+2} n)^{k-\varepsilon}) = O(n^{k-\varepsilon'})$ time algorithm for solving #ColHS, for any $\varepsilon' < \varepsilon$, thus refuting #SETH.

References

[1] A. Abboud, A. Backurs, and V. V. Williams. Quadratic-time hardness of LCS and other sequence similarity measures. FOCS 2015.

[2] A. Abboud and V. V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. In FOCS 2014, pages 434–443.

[3] A. Abboud, V. V. Williams, and O. Weimann. Consequences of faster alignment of sequences. In ICALP 2014, pages 39–51.

[4] A. Backurs and P. Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In STOC 2015, pages 51–58.

[5] D. Bienstock. Graph searching, path-width, tree-width and related problems (a survey). In *Reliability of computer and communication networks (New Brunswick, NJ, 1989)*, volume 5 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 33–49. Amer. Math. Soc., Providence, RI, 1991.

[6] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Fourier meets Möbius: fast subset convolution. In STOC 2007, pages 67–74.

[7] M. Bläser and H. Dell. Complexity of the cover polynomial. In ICALP 2007, pages 801–812.

[8] H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. In ICALP 2013, pages 196–207.

[9] G. Borradaile and H. Le. Optimal dynamic program for r-domination problems over tree decompositions. *CoRR*, abs/1502.00716, 2015.

[10] K. Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In FOCS 2014, pages 661–670.

[11] K. Bringmann and M. Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. To appear in FOCS 2015.

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

[12] H. Broersma, P. A. Golovach, and V. Patel. Tight complexity bounds for FPT subgraph problems parameterized by the clique-width. *Theor. Comput. Sci.*, 485:69–84, 2013.

[13] J. Cai and V. Choudhary. Valiant's Holant theorem and matchgate tensors. *Theor. Comput. Sci.*, 384(1):22–32, 2007.

[14] J. Cai and A. Gorenstein. Matchgates revisited. *Theory of Computing*, 10:167–197, 2014.

[15] J. Cai, H. Guo, and T. Williams. The complexity of counting edge colorings and a dichotomy for some higher domain Holant problems. In FOCS 2014, pages 601–610.

[16] J. Cai, S. Huang, and P. Lu. From Holant to #CSP and back: Dichotomy for Holant^c problems. *Algorithmica*, 64(3):511–533, 2012.

[17] J. Cai, P. Lu, and M. Xia. Holant problems and counting CSP. In STOC 2009, pages 715–724.

[18] J. Cai, P. Lu, and M. Xia. Computational complexity of Holant problems. *SIAM J. Comput.*, 40(4):1101–1132, 2011.

[19] J. Cai, P. Lu, and M. Xia. Dichotomy for Holant* problems of Boolean domain. In SODA 2011, pages 1714–1728.

[20] J. Cai, P. Lu, and M. Xia. Dichotomy for Holant* problems with domain size 3. In SODA 2013, pages 1278–1295.

[21] J.-Y. Cai and P. Lu. Holographic algorithms: From art to science. In STOC 2007, pages 401–410.

[22] J.-Y. Cai, P. Lu, and M. Xia. Holographic algorithms by Fibonacci gates and holographic reductions for hardness. In FOCS 2008, pages 644–653.

[23] C. Calabro and R. Paturi. k-Sat is no harder than Decision-Unique-k-Sat. In CSR 2009, pages 59–70.

[24] R. H. Chitnis, M. Hajiaghayi, and D. Marx. Tight bounds for Planar Strongly Connected Steiner Subgraph with fixed number of terminals (and extensions). In *SODA 2014*, pages 1782–1801, 2014.

[25] B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

[26] B. Courcelle. The monadic second-order logic of graphs III: Treewidth, forbidden minors and complexity issues. *Informatique Théorique*, 26:257–286, 1992.

[27] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000.

[28] B. Courcelle, J. A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1-2):23–52, 2001.

[29] B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

[30] R. Curticapean. Block interpolation: A framework for tight exponential-time counting complexity. In ICALP 2015, pages 380–392.

[31] R. Curticapean. *The simple, little and slow things count: on parameterized counting complexity*. PhD thesis, Universität des Saarlandes, Postfach 151141, 66041 Saarbrücken, 2015.

[32] R. Curticapean and M. Xia. Parameterizing the permanent: Genus, apices, minors, evaluation mod 2^k. To appear in FOCS 2015.

[33] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. On problems as hard as CNF-SAT. In CCC 2012, pages 74–84.

[34] M. Cygan, S. Kratsch, and J. Nederlof. Fast hamiltonicity checking via bases of perfect matchings. In STOC 2013, pages 301–310.

[35] M. Cygan, D. Marx, M. Pilipczuk, and M. Pilipczuk. Hitting forbidden subgraphs in graphs of bounded treewidth. In MFCS 2014, pages 189–200.

[36] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, and J. O. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In FOCS 2011, pages 150–159.

[37] H. Dell, T. Husfeldt, D. Marx, N. Taslaman, and M. Wahlen. Exponential time complexity of the permanent and the Tutte polynomial. *ACM Transactions on Algorithms*, 10(4):21:1–21:32, 2014.

[38] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Bidimensional parameters and local treewidth. *SIAM J. Discrete Math.*, 18(3):501–511, 2004.

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

[39] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Fixed-parameter algorithms for (k,r)-Center in planar graphs and map graphs. *ACM Transactions on Algorithms*, 1(1):33–47, 2005.

[40] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005.

[41] E. D. Demaine and M. Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51(3):292–302, 2008.

[42] E. D. Demaine and M. Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008.

[43] E. D. Demaine and M. T. Hajiaghayi. Fast algorithms for hard graph problems: Bidimensionality, minors, and local treewidth. In *Graph Drawing*, pages 517–533, 2004.

[44] F. Dorn, F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh. Beyond bidimensionality: Parameterized subexponential algorithms on directed graphs. In *STACS 2010*, pages 251–262, 2010.

[45] F. Dorn, F. V. Fomin, and D. M. Thilikos. Subexponential parameterized algorithms. *Computer Science Review*, 2(1):29–39, 2008.

[46] F. Dorn, E. Penninkx, H. L. Bodlaender, and F. V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010.

[47] J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[48] W. Espelage, F. Gurski, and E. Wanke. How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In WG 2001, pages 117–128.

[49] M. R. Fellows, F. A. Rosamond, U. Rotics, and S. Szeider. Clique-width is NP-complete. *SIAM J. Discrete Math.*, 23(2):909–939, 2009.

[50] F. V. Fomin, P. A. Golovach, D. Lokshtanov, and S. Saurabh. Intractability of clique-width parameterizations. *SIAM J. Comput.*, 39(5):1941–1956, 2010.

[51] F. V. Fomin, P. A. Golovach, D. Lokshtanov, and S. Saurabh. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM J. Comput.*, 43(5):1541–1563, 2014.

[52] F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh. Subexponential algorithms for partial cover problems. *Inf. Process. Lett.*, 111(16):814–818, 2011.

[53] F. V. Fomin, D. Lokshtanov, and S. Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In SODA 2014, pages 142–151.

[54] F. V. Fomin and D. M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM J. Comput.*, 36(2):281–309, 2006.

[55] A. Galluccio and M. Loebl. On the theory of Pfaffian orientations. I. Perfect matchings and permanents. *Electr. J. Comb.*, 6, 1999.

[56] M. U. Gerber and D. Kobler. Algorithms for vertex-partitioning problems on graphs with fixed clique-width. *Theor. Comput. Sci.*, 1-3(299):719–734, 2003.

[57] O. Giménez, P. Hlinený, and M. Noy. Computing the Tutte polynomial on graphs of bounded clique-width. *SIAM J. Discrete Math.*, 20(4):932–946, 2006.

[58] B. Godlin, T. Kotek, and J. A. Makowsky. Evaluations of graph polynomials. In WG 2008, pages 183–194.

[59] H. Guo, P. Lu, and L. G. Valiant. The complexity of symmetric Boolean parity Holant problems. *SIAM J. Comput.*, 42(1):324–356, 2013.

[60] S. Huang and P. Lu. A dichotomy for real weighted Holant problems. In CCC 2012, pages 96–106.

[61] R. Impagliazzo and R. Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[62] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Computer and System Sciences*, 63(4):512–530, 2001.

[63] P. W. Kasteleyn. The statistics of dimers on a lattice: I. The number of dimer arrangements on a quadratic lattice. *Physica*, 27(12):1209–1225, 1961.

[64] P. W. Kasteleyn. Graph theory and crystal physics. In *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, London, 1967.

[65] P. N. Klein and D. Marx. Solving Planar k-Terminal Cut in $O(n^{c\sqrt{k}})$ time. In *ICALP 2012 (1)*, pages 569–580.

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.

[66] P. N. Klein and D. Marx. A subexponential parameterized algorithm for Subset TSP on planar graphs. In *SODA 2014*, pages 1812–1830.

[67] D. Kobler and U. Rotics. Polynomial algorithms for partitioning problems on graphs with fixed clique-width (extended abstract). In SODA 2001, pages 468–476.

[68] D. Kobler and U. Rotics. Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Applied Mathematics*, 126(2-3):197–221, 2003.

[69] M. Kowalczyk and J. Cai. Holant problems for regular graphs with complex edge functions. In STACS 2010, pages 525–536.

[70] J. M. Landsberg. *Tensors: geometry and applications*, volume 128 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2012.

[71] D. Lokshtanov, D. Marx, and S. Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In SODA 2011, pages 777–789.

[72] D. Lokshtanov, D. Marx, and S. Saurabh. Slightly superexponential parameterized problems. In SODA 2011, pages 760–776.

[73] J. A. Makowsky, U. Rotics, I. Averbouch, and B. Godlin. Computing graph polynomials on graphs of bounded clique-width. In WG 2006, pages 191–204.

[74] V. Y. Pan. Simple multivariate polynomial multiplication. *J. Symb. Comput.*, 18(3):183–186, 1994.

[75] M. Patrascu and R. Williams. On the possibility of faster SAT algorithms. In SODA 2010, pages 1065–1075.

[76] M. Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In MFCS 2011, pages 520–531.

[77] M. Pilipczuk, M. Pilipczuk, P. Sankowski, and E. J. van Leeuwen. Subexponential-time parameterized algorithm for Steiner Tree on planar graphs. In *STACS 2013*, pages 353–364.

[78] M. Pilipczuk, M. Pilipczuk, P. Sankowski, and E. J. van Leeuwen. Network sparsification for Steiner problems on planar and bounded-genus graphs. In *FOCS 2014*, pages 276–285.

[79] M. Rao. MSOL partitioning problems on graphs of bounded treewidth and clique-width. *Theor. Comput. Sci.*, 377(1-3):260–267, 2007.

[80] T. Regge and R. Zecchina. Combinatorial and topological approach to the 3d ising model. *Journal of Physics A: Mathematical and General*, 33(4):741–761, 2000.

[81] K. Suchan and I. Todinca. On powers of graphs of bounded NLC-width (clique-width). *Discrete Applied Mathematics*, 155(14):1885–1893, 2007.

[82] H. N. V. Temperley and M. E. Fisher. Dimer problem in statistical mechanics-an exact result. *Philosophical Magazine*, 6(68):1061–1063, 1961.

[83] G. Tesler. Matchings in graphs on non-orientable surfaces. *J. Comb. Theory, Ser. B*, 78(2):198–231, 2000.

[84] D. M. Thilikos. Fast sub-exponential algorithms and compactness in planar graphs. In *ESA 2011*, pages 358–369.

[85] L. G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[86] L. G. Valiant. Holographic algorithms (extended abstract). FOCS 2004, pages 306–315.

[87] L. G. Valiant. Holographic algorithms. *SIAM J. Comput.*, 37(5):1565–1594, 2008.

[88] J. M. M. van Rooij, H. L. Bodlaender, and P. Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In ESA 2009, pages 566–577.

[89] E. Wanke. *k*-NLC graphs and polynomial algorithms. *Discrete Applied Mathematics*, 54(2-3):251–266, 1994.

[90] R. Williams and H. Yu. Finding orthogonal vectors in discrete structures. In SODA 2014, pages 1867–1877.

Copyright © by SIAM.
Unauthorized reproduction of this article is prohibited.